

**NASA  
Technical  
Memorandum**

**NASA TM - 100364**

**A METHODOLOGY FOR COMMONALITY ANALYSIS,  
WITH APPLICATIONS TO SELECTED  
SPACE STATION SYSTEMS**

**By Lawrence Dale Thomas**

**Space Station Project Engineering Office  
Science and Engineering Directorate**

**May 1989**

**{NASA-TM-100364} A METHODOLOGY FOR  
COMMONALITY ANALYSIS, WITH APPLICATIONS TO  
SELECTED SPACE STATION SYSTEMS {NASA.  
Marshall Space Flight Center} 99 p CSCL 22B**

**N89-24421**

**Unclas  
G3/18 0214649**



**National Aeronautics and  
Space Administration**

**George C. Marshall Space Flight Center**



1. REPORT NO. NASA TM-100364		2. GOVERNMENT ACCESSION NO.		3. RECIPIENT'S CATALOG NO.	
4. TITLE AND SUBTITLE A Methodology for Commonality Analysis, with Application to Selected Space Station Systems				5. REPORT DATE May 1989	
				6. PERFORMING ORGANIZATION CODE EJ13	
7. AUTHOR(S) L. Dale Thomas				8. PERFORMING ORGANIZATION REPORT #	
9. PERFORMING ORGANIZATION NAME AND ADDRESS  George C. Marshall Space Flight Center Marshall Space Flight Center, Alabama 35812				10. WORK UNIT NO.	
				11. CONTRACT OR GRANT NO.	
12. SPONSORING AGENCY NAME AND ADDRESS  National Aeronautics and Space Administration Washington, D.C. 20546				13. TYPE OF REPORT & PERIOD COVERED  Technical Memorandum	
				14. SPONSORING AGENCY CODE	
15. SUPPLEMENTARY NOTES Prepared by Space Station Project Engineering Office, Science and Engineering Directorate.					
16. ABSTRACT  The application of commonality in a system represents an attempt to reduce costs by reducing the number of unique components. A formal method for conducting commonality analysis has not been established. In this dissertation, commonality analysis is characterized as a partitioning problem. The cost impacts of commonality are quantified in an objective function, and the solution is that partition which minimizes this objective function. Clustering techniques are used to approximate a solution, and sufficient conditions are developed which can be used to verify the optimality of the solution. This method for commonality analysis is general in scope. It may be applied to the various types of commonality analysis required in the Conceptual, Preliminary, and Detail Design phases of the system development cycle.					
17. KEY WORDS System Engineering Commonality Analysis Hierarchical Clustering Applications			18. DISTRIBUTION STATEMENT  Unclassified - Unlimited		
19. SECURITY CLASSIF. (of this report) Unclassified		20. SECURITY CLASSIF. (of this page) Unclassified		21. NO. OF PAGES 102	
				22. PRICE NTIS	



# TABLE OF CONTENTS

	Page
I. INTRODUCTION .....	1
A. General .....	1
B. Background .....	1
C. The Scope of Commonality Analysis .....	2
D. Application of Cluster Analysis .....	2
II. DEFINITION OF THE COMMONALITY ANALYSIS PROBLEM .....	3
A. General .....	3
B. Economic Considerations of Commonality .....	3
C. The Number of Commonality Alternatives .....	5
D. Economic Feasibility of Commonality Alternatives .....	7
III. THE APPLICABILITY OF CLUSTERING METHODS .....	8
A. Relevant Applications Utilizing Cluster Analysis .....	8
B. Variance Based Clustering Methods .....	9
C. Optimality Considerations .....	10
IV. A CLUSTERING BASED COMMONALITY ANALYSIS METHODOLOGY .....	11
A. Specification of an Object .....	11
B. Specification of a Common Item .....	13
C. Selection of Commonality Alternatives .....	16
D. Optimality Considerations in the Selection of Commonality Alternatives .....	18
E. S-Minimal Partitions .....	19
F. Z-Minimal Partitions .....	20
G. Illustration of Methodology .....	23
V. COMMONALITY ANALYSIS AND THE SYSTEM DEVELOPMENT CYCLE ....	30
A. General .....	30
B. The System Design Process .....	30
C. The Role of Commonality Analysis in Conceptual Design .....	31
D. Commonality Analysis of Space Station Module Berthing Interfaces .....	32
E. The Role of Commonality Analysis in Preliminary Design .....	41
F. Commonality Analysis of Electric Motors (Part 1) .....	42
G. The Role of Commonality Analysis in Detail Design .....	47
H. Commonality Analysis of Electric Motors (Part 2) .....	48
VI. CONCLUSIONS AND RECOMMENDATIONS .....	51
A. Conclusions .....	51
B. Recommendations .....	52

## TABLE OF CONTENTS (Concluded)

	Page
APPENDIX A. SELECTED PROOFS AND ILLUSTRATIONS .....	57
APPENDIX B. COMPUTER PROGRAMS FOR COMMONALITY ANALYSIS .....	65
APPENDIX C. APPLICATION OF SUFFICIENCY CONDITIONS TO PHASE C/D COMMONALITY ANALYSIS CLUSTERING SOLUTION .....	91
REFERENCES .....	93

## LIST OF ILLUSTRATIONS

Figure	Title	Page
1.	Hierarchical clustering of berthing interfaces .....	38
2.	Variation of excess functionality with number of berths .....	41
3.	Hierarchical clustering of electric motors.....	44
4.	Variation of excess functionality with number of types of motors .....	46
5.	Cost benefit of various levels of commonality.....	50

# LIST OF TABLES

Table	Title	Page
1.	Number of Commonality Alternatives for 40 Items or Less .....	6
2.	Space Station Water Tanks.....	12
3.	Initial Similarity Matrix for Space Station Water Tanks (\$M).....	24
4.	Second Similarity Matrix for Space Station Water Tanks (\$M).....	25
5.	Final Similarity Matrix for Space Station Water Tanks (\$M).....	25
6.	Modified Similarity Matrix for Space Station Water Tanks (\$M).....	26
7.	Increase in Excess Functionality [ $\delta(u,v)$ ] from the Clustering of Two Tanks .....	28
8.	Maximum Cost Savings from Switching Tanks Between Clusters.....	28
9.	Excess Functionality of Clusters in Clustering Solution.....	28
10.	Z-Minimal Partitions over Range of N .....	29
11.	Berthing Interface Functions .....	33
12.	Module Berthing Interfaces .....	34
13.	Hierarchically Formed Partition (K=30) .....	39
14.	Final Clustering Solution .....	40
15.	Preliminary Design Data for Electric Motors .....	43
16.	Hierarchically Formed Partition (K=300).....	45
17.	Detail Design Data for Electric Motors .....	49
18.	Hierarchically Formed Partition .....	50



## TECHNICAL MEMORANDUM

### A METHODOLOGY FOR COMMONALITY ANALYSIS, WITH APPLICATIONS TO SELECTED SPACE STATION SYSTEMS

#### I. INTRODUCTION

##### A. General

The life-cycle cost of systems is typically comprised of four components: research and development (R&D) cost, production cost, operation cost, and disposal cost [1]. Because of the typically small production lot sizes and state-of-the-art technology applications, the R&D costs are disproportionately large for aerospace systems relative to commercial systems development programs. Commonality seeks to reduce R&D costs by replacing duplicate R&D efforts on functionally similar system components with a single development effort, thereby reducing R&D costs. Likewise, commonality seeks to reduce production costs through larger production lot sizes, and to reduce operations costs through increased standardization.

The R&D cost and the production cost, when summed, give the purchase cost of a system. While production costs in general vary proportionately with the production lot size, R&D costs are in general nonrecurring; they are incurred only once whether the production lot size is one or one hundred. Generally, for space systems, two common items may be purchased for 60 to 65 percent of the cost incurred when purchasing two uniquely developed items which perform the same functions [2]. In addition to purchase costs, operations costs can also be reduced through the application of commonality - the Boeing Company developed a common cockpit for both its 757 and 767 airplanes based on the operations cost savings which the buyer would realize as a result of reduced pilot training [3].

The cost saving potential of commonality in the space station program was recognized in the early stages of the program. The conceptual design of the space station included five modules: two laboratories, two living quarters, and a logistics module. In a study described by Powell and Beam [4], a common module was defined which could be outfitted to serve as a laboratory, living quarters, or a logistics module. This commonality analysis was performed largely in trial and error fashion, with engineering intuition providing the definition of commonality options. The cost of each option was evaluated, and the option having the lowest cost became the common module concept.

##### B. Background

Commonality analysis is a routine procedure during the early phases of the system development cycle in aerospace programs. In most cases, the number of commonality alternatives is quite small and all alternatives may be explored to determine the optimum. The 757/767 common cockpit is such an example - there are only two alternatives: (1) develop a unique cockpit for each airplane or (2) develop a common cockpit for both airplanes. However, there are cases such as the common module study described above in which the number of alternatives is tremendous. In these cases, it is not practical to examine all alternatives and an ad hoc approach must be employed.

A lack of knowledge regarding a method for dealing with the large number of commonality alternatives and a lack of understanding of both the economics of commonality and its evolution over the system development cycle have combined to make "commonality" a hated word for many project managers. However, a recent study [2] addressed the economic impacts of commonality. The research documented in this report represents an initial attempt to attack the other two problems: (1) the scope of commonality analysis in the various design phases, and (2) a method to cope with the large number of commonality alternatives.

### C. The Scope of Commonality Analysis

The role of commonality is basically one of establishing the set of requirements which an item must satisfy from the systems standpoint rather than from the subsystems or even lower standpoints. As such, the integration of commonality analysis into the system development cycle is fairly straightforward; the results of commonality analysis must track the overall evolution of design requirements in terms of both schedule and detail. The objectives of commonality analysis evolve as the system itself evolves. A commonality analysis performed during Conceptual Design differs markedly from one performed during Detail Design. Early studies address the cost effectiveness of commonality and its sensitivity to parametric variation. Follow-on studies in later design phases address the commonality solution - what functions a common item should include and where it should be used. Section V discusses in detail the scope of commonality in each of the four phases of the system development cycle.

### D. Application of Cluster Analysis

In Section II, commonality analysis is characterized as a partitioning problem, with the objective being to find the partition which minimizes cost. The cost in this case is life cycle cost. Life cycle cost best quantifies the cost/performance tradeoff implicit in commonality. While commonality decreases costs such as R&D, the physical manifestations can be increased weight, volume, power consumption, or other system variables. If these costs may be described by an objective function, clustering techniques may be used to identify a partition which is optimum or near optimum. A large number of cluster analysis algorithms are currently defined and in use. Section III describes the selection of the best cluster analysis method for use in commonality analysis.

Section IV presents the formulation of a cluster analysis-based methodology for the performance of commonality analysis. Given a set of functionally similar items, such as a set of water tanks of different sizes, cluster analysis may be employed to specify whether any tanks are similar enough to be developed in common. Each item is characterized in terms of its design specifications and required functions by an attribute vector. Feasibility relations are used to define the attributes of a common item in terms of its constituents; if the value of the objective function decreases from the use of a common item instead of the constituent items, those items are clustered. Once the clustering is completed, each cluster defines a common item. Since clustering techniques do not inherently guarantee generation of an optimum solution, sufficient conditions which may be used in the verification of optimum solutions are developed. These conditions are presented in Section IV. This commonality analysis methodology is illustrated in three examples in Section V using space station program design data.

Finally, conclusions and recommendations are presented in Section VI.

## II. DEFINITION OF THE COMMONALITY ANALYSIS PROBLEM

### A. General

The general formulation of commonality is one in which a new entity AB results from making item A common with item B. This new entity AB incorporates the functions required of both A and B. For example, if pump A must operate under high pressure and pump B must provide a high flow rate, then a common pump AB must operate under high pressure and/or provide high flow rates. By allowing the creation of new items incorporating the functions of any two or more items, any degree of commonality is technically feasible within system performance constraints, and the discriminator in feasibility is one of economy.

### B. Economic Considerations of Commonality

The System Commonality Analysis Tool (SCAT) is a computer program previously developed to aid in the evaluation of the cost impacts associated with the application of commonality to spacecraft development programs [2]. In general, it is not applicable to commonality analyses involving more than two items, but the program does effectively quantify the cost and performance impacts associated with commonality between two items. These economic relationships are reviewed subsequently and will serve as a basis for quantifying the economic impacts of commonality in the methodology to be developed.

The SCAT assesses the economic impact of commonality on the basis of Life Cycle Cost (LCC). LCC is determined as a sum of the purchase cost – R&D plus production – and operations cost:

$$LCC = C_p + C_o$$

where  $C_p$  is purchase cost and  $C_o$  is operations cost. These costs are not discounted; discounting may be included by manually discounting the component costs of  $C_p$  and  $C_o$ . Retirement cost is assumed equal for all commonality alternatives and, thus, is not included in the analysis. The purchase costs provide an accounting of the differences between the R&D and production costs with various commonality alternatives. Operations costs provide an accounting of differences in performance levels; differences in energy usage, weight, and maintenance costs impact operating costs rather than purchase costs.

The study leading to the development of the SCAT identified ten attributes of a hardware item which determine the cost effectiveness of commonality:

1. Quantity (QN)
2. Weight (WGT)

3. Volume (VOL)
4. Power Consumption (PWR)
5. Duty Cycle (DC)
6. Mean Time Between Failure (MTBF)
7. Mean Time To Repair (MTTR)
8. Maintenance Environment (ME)
9. Research and Development Cost (RDC)
10. First Item Production Cost (FPC).

While other attributes relating to an item's function are needed to establish technical feasibility of a commonality alternative, these ten attributes are the principle contributors to the economic feasibility. Each of these attributes can assume only non-negative values.

The purchase cost impact of commonality is the result of eliminating duplicative R&D efforts and of lowering unit production cost through larger production lot quantities. For M items, the purchase cost is given as

$$C_p = \sum_{i=1}^M [RDC_i + (\text{Production Cost})_i + (\text{Launch Cost})_i]$$

where

$$(\text{Production Cost})_i = f(\text{FPC}_i, \text{QN}_i) \quad ,$$

$$(\text{Launch Cost})_i = f(\text{WGT}_i, \text{QN}_i) \quad .$$

Commonality affects operations costs by changes, within technical constraints, to system performance variables. These changes result from the excess functionality of common items and of the reduced number of spares required onboard. For M items, the operations cost is given as:

$$C_o = \sum_{i=1}^M (\text{VC}_i + \text{SPARVC}_i + \text{EC}_i + \text{SPROD}_i + \text{STRANC}_i + \text{REPC}_i + \text{THERC}_i)$$

where

$$\text{Volume Cost} = \text{VC}_i = f(\text{VOL}_i, \text{QN}_i) \quad ,$$

$$\text{Volume Cost of Spares} = \text{SPARVC}_i = f(\text{VOL}_i) \quad ,$$

$$\text{Energy Cost} = \text{EC}_i = f(\text{DC}_i, \text{QN}_i, \text{PWR}_i) \quad ,$$

$$\text{Production Cost of Spares} = \text{SPROD}_i = f(\text{FPC}_i, \text{QN}_i) \quad ,$$

$$\text{Launch Cost of Spares} = \text{STRANC}_i = f(\text{WGT}_i, \text{VOL}_i, \text{MTTR}_i, \text{MTBF}_i, \text{DC}_i, \text{QN}_i) \quad ,$$

$$\text{Replacement Cost} = \text{REPC}_i = f(\text{MTTR}_i, \text{MTBF}_i, \text{DC}_i, \text{QN}_i) \quad ,$$

$$\text{Thermal Cost} = \text{THERC}_i = f(\text{DC}_i, \text{QN}_i, \text{PWR}_i) \quad .$$

All of the foregoing functions are strictly increasing over the domain of their respective variables. Two exceptions are the launch cost of spares and replacement cost; they are strictly decreasing over the domain of MTBF and strictly increasing over the domain of all other variables. A detailed derivation of these cost relationships is given in the user's manual for the SCAT [5].

### C. The Number of Commonality Alternatives

In a commonality analysis including two items A and B, there are two commonality alternatives: (1) build both item A and item B, which is the "no commonality" alternative; and (2) build item AB to serve in the applications of former items A and B, which is the "total commonality" option. This is the form of commonality analysis for which the SCAT was developed [2]. When more than two items are included in a commonality analysis, the number of commonality alternatives which must be evaluated grows significantly. Consider a commonality analysis including three items A, B, and C. The commonality alternatives which must be evaluated include:

- (1) item A, item B, and item C, (no commonality)
- (2) item AB and item C, (partial commonality)
- (3) item AC and item B, (partial commonality)
- (4) item BC and item A, (partial commonality)
- (5) item ABC (total commonality).

In the case of four items, the number of commonality alternatives grows to 15 and, for five items, to 52 alternatives. In all cases, all commonality alternatives but two are "partial commonality" alternatives.

The enumeration of commonality alternatives is analogous to evaluating the partitions of M objects into N groups over all values of N from one to M. The number of ways of sorting M objects into N groups is a Stirling number of the second kind [6]:

$$S_M(N) = (1/N!) \cdot \sum_{k=0}^N (-1)^{N-k} \cdot \{N!/[k! \cdot (N-k)!]\} \cdot k^M \quad .$$

There are 2,436,684,974,110,751 possible partitions of 25 objects into five groups. If the number of groups is varied from one to M, the number of possibilities is then a sum of Stirling numbers:

$$S_M = \sum_{N=1}^M S_M(N) \quad .$$

In the case of 25 observations, there are more than  $4 \times 10^{18}$  possible partitions of any size ( $1 \leq N \leq 25$ ). The total number of commonality alternatives for  $M \leq 40$  is given in Table 1.

TABLE 1. NUMBER OF COMMONALITY ALTERNATIVES FOR  
40 ITEMS OR LESS

<u>Number of Items</u>	<u>Number of Commonality Alternatives</u>
1	-
2	2
3	5
4	15
5	52
6	203
7	877
8	4,140
9	21,147
10	115,975
15	1,382,958,545
20	$5.2 \times 10^{13}$
25	$4.6 \times 10^{18}$
30	$8.5 \times 10^{23}$
35	$2.8 \times 10^{29}$
40	$1.6 \times 10^{35}$

For commonality analysis, the objective is to determine the optimal partition with regard to cost. Given an objective function which will be minimum when an optimal partition of M objects into N groups with regard to cost has been achieved, the most straightforward solution is to evaluate all possible partitions.

In theory, of course, the problem is simple; to quote Dr. Idnozo Hcahscror-Tenib, that super galactian hypermetrician who appeared in Thorndike's 1953 Presidential address to the Psychometric society, "Is easy. Finite number of combinations. Only 563 billion billion billion. Try all. Keep best." [7]

While complete enumeration guarantees generation of an optimum solution, it is only applicable to small sets of objects. If a supercomputer could evaluate one million partitions per second, more than one year would be needed to find the optimal partition of 25 objects into five groups, and it would take approximately two thousand years to determine the optimum partition of any size by exhaustive search.

#### D. Economic Feasibility of Commonality Alternatives

Recall that when item A is made common with item B, the new item AB must satisfy the functional requirements of both item A and item B. In general, item AB will then possess some excess functions when compared to item A and other excess functions compared to item B. The manifestations of this excess functionality can be increased weight, volume, and power consumption or decreased MTBF for hardware items; another manifestation can be increased complexity of the item AB relative to items A and B, resulting in increased purchased cost per unit.

When viewing commonality as a partitioning problem, the overall system life cycle costs previously discussed can be reduced in one of two ways: (1) reduction of the number of unique types of items N, and (2) reduction of the quantity of excess functionality S. Clearly, for two items A and B which are functionally identical, cost savings result from making A and B common; cost savings will follow from the elimination of a redundant R&D effort and from a single production lot instead of two smaller production lots. However, in general, items A and B are only similar rather than identical, and the analysis must address the cost effects associated with excess functionality.

Recalling the various cost impacts of commonality discussed in a previous section, these costs affected by N are strictly increasing in N. Given M functionally similar types of items partitioned into N groups, the overall quantity of items, including spares, is less than or equal to the overall quantity for M items partitioned into M groups. Then, although commonality increases individual production lot sizes, the net quantity over all production lots either decreases or remains constant. Thus, the production cost for N types of items remains constant or, in the presence of a learning curve, is strictly increasing in N. The spares production cost is then also constant or strictly increasing in N. Fewer types of unique items N mean fewer spares which must be stored onboard, reducing spares volume cost; thus, spares volume cost is strictly increasing in N. If all  $RDC_i$  are nonzero, then both the purchase cost and operations cost are strictly increasing in N.

Likewise, as commonality results in fewer unique types of items, some or all of the remaining types of items possess some excess functionality. This excess functionality S manifests itself in terms of one or more the following: increased item

weight, volume, power consumption, MTTR, R&D cost, or first item production cost, or reduced MTBF. The cost functions will increase for these changes in the variables, as previously discussed.

Unlike N, neither the purchase cost nor the operations cost is strictly increasing in S in general. For a particular commonality alternative, any increase in excess functionality S necessarily results in increased cost. However, for a given level of commonality N there will be any number of commonality alternatives, each with an associated and, in general, distinct value of S. Since some commonality applications result in greater cost savings than others, more excess functionality S may be associated with greater overall cost effectiveness.

To illustrate the relationship of N and S, consider the development of three functionally similar items A, B, and C. Their development requires three research and development efforts ( $N=M=3$ ), three production lots, three types of spares stored on board, and given levels of launch cost, volume cost, energy cost, etc. The development of a common item ABC for all three applications requires only one development effort, and the production lot size is the sum of the former lot sizes for items A, B, and C ( $N=1, M=3$ ). In the presence of a learning curve, even small increases in small production lot sizes can significantly lower unit production costs. Thus, cost savings on the order of two R&D efforts plus some reduction in production costs will result. For nonidentical items A, B, and C, some level of excess functionality  $S > 0$  will result from the development of common item ABC. If the excess functionality S of item ABC is small enough that the increases in launch cost, volume cost, energy cost, etc., are smaller than the associated cost savings, then common item ABC is cost effective.

Before deciding to implement common item ABC in place of items A, B, and C, other commonality alternatives should be evaluated. In this case, common item AB and item C, common item AC and item B, and common item BC and item A are the remaining commonality alternatives. While the economic and technical factors governing the advantageous application of commonality are understood from the development of the SCAT, the method by which the entire solution space should be evaluated is not established.

### III. THE APPLICABILITY OF CLUSTERING METHODS

#### A. Relevant Applications Utilizing Cluster Analysis

While a formal method for commonality analysis has not been established, a review of relevant literature does indicate a direction. A problem similar to commonality analysis is that of defining machine-component groupings in group technology. Burbidge [8], one of the group technology pioneers, defined a method known as "production flow analysis" to identify machine-component groupings. However, because his production flow analysis was manual and largely heuristic in nature, it was not widely applied. Later, McAuley [9] described a method for machine-component grouping based on a single linkage cluster analysis approach. His approach was based on a simple clustering technique, was relatively easy to implement on a computer, and generated consistent solutions. Recent research in the machine-component grouping area [10,11,12] consists of variations of McAuley's clustering approach.



A study which, in retrospect, may be categorized as commonality analysis is described by McCormick, et al. [13]. He used a cluster analysis to assess the commonality of the various aircraft within the Air Force. His study examined the functions which could be performed by each aircraft and the applications required of military aircraft. The result of his analysis was to group the aircraft into five families with each member of a family having similar functionality. Likewise, the applications were also grouped by similarity, where their similarity was defined as being able to be performed by the same aircraft. A similar study of Army tactical weapons and their functions was conducted by Perrin [14].

Yeager [15] specifically addresses commonality and formulates commonality analysis as a partitioning problem, with the objective being to find the partition which minimizes an objective function. He subsequently proceeds to develop the mathematical framework for solution heuristics for a special case of commonality analysis. In the case where the functions required of item A dominate those required by item B, item A may substitute for item B rather than requiring a new item AB. He specifies several types of relations which may be defined for a data set based on the feasibility of these substitutions. These relations serve as the basis for a branch-and-bound approach to pare the solution space to a level where the optimum solution may be determined by exhaustive evaluation of the remaining alternatives. One such relation is a joining relation whereby two objects are grouped in a single partition to improve the objective function. Yeager suggests use of the joining relation in an algorithm analogous to a clustering method.

## B. Variance Based Clustering Methods

The foregoing discussion suggests that clustering methods may be applicable to commonality analysis. Still, from the plethora of clustering methods available, the selection of the most appropriate clustering method constitutes another problem. Mojena [16] and Valev [17] noted the frequent use of hierarchical clustering methods in partitioning applications. Mojena observed that these methods tend to produce good, although not necessarily optimal, partitions. His study included a comparative assessment of the several hierarchical methods based on Monte Carlo simulation, and results indicated that variance methods performed best over the widest range of data sets. Similar results were generated by Blashfield [18] and Milligan et al. [19].

Variance methods hierarchically build clusters in the manner which successively minimizes the variance within each cluster, in effect minimizing the loss of information involved in the clustering process [20]. Variance methods are very flexible in nature and can be used to minimize an objective function in an analogous manner to minimizing variance; the clusters are hierarchically formed in a manner to successively minimize the objective function. The steps involved in use of the variance algorithm are as follows:

- 1) Begin with N clusters of one object each. Let the clusters be labeled 1 through N.
- 2) Determine the N x N lower triangular similarity (distance) matrix to document the distances between all objects. The distance here,  $D(u,v)$ , represents the change in the objective function which will be realized if objects u and v are clustered.
- 3) Search the similarity matrix for the most similar pair of objects [minimum  $D(u,v)$ ]. Let these clusters be labeled p and q, with  $p < q$ .

- 4) Reduce the number of clusters,  $N$ , by one through the merger of clusters  $p$  and  $q$ . Label the product of the merger  $p$  and remove the row and column of the similarity matrix pertaining to cluster  $q$ .
- 5) Repeat steps (2) through (4) for the remaining clusters, which will again result in one less cluster. Continue repeating steps (2) through (4), each time reducing the number of clusters by one, until no further mergers are possible which will reduce the objective function or, if a variance measure is used, until all  $M$  objects constitute a single cluster.

Occasionally a tie situation may be encountered in step (3), with two or more pairs of clusters having identically minimal distances  $D(u,v)$ . Zupan [21] notes that the most widely implemented tie resolution strategy is the selection of the first or last pair of clusters having the same minimal distance, while Romesburg [22] suggests a random selection strategy in the event of a tie.

### C. Optimality Considerations

A problem encountered when clustering by hierarchical methods is that clusters formed in the early stages of the analysis may lead to sub-optimal partitions in later stages [7]. This is attributable to the fact that a hierarchical clustering scheme imposes the ultrametric inequality on the distances  $D(u,v)$  in a similarity matrix. The ultrametric inequality

$$D(u,v) \leq \max [D(u,w), D(v,w)]$$

was defined by Johnson [23] and requires, for any three objects  $A$ ,  $B$ , and  $C$  in a cluster analysis, that

$$D(AB,C) = D(A,C) = D(B,C) \quad .$$

The ultrametric inequality requires that the two objects or clusters  $u$  and  $v$  which are separated by the minimum distance  $D(u,v)$  are each equally distant from all other objects or clusters. Once clustered, the cluster  $uv$  and another object  $w$ , which are separated by the minimum distance  $D(uv,w)$ , are then each equally distant from all other objects. The ultrametric inequality is a sufficient condition for optimality in that if it holds for each stage of the hierarchical clustering process, it guarantees an optimum solution. In general, a similarity matrix will not satisfy the ultrametric inequality, and Johnson uses the condition only as a basis to derive two clustering methods equivalent to the single and complete linkage techniques.

A dynamic programming approach developed by Jensen [24] generates globally optimum solutions, but the algorithm is quite computationally intensive. While it does reduce the solution space by orders of magnitude from that required by total enumeration of partitions, the computational burden is excessive for problems involving more than 12 to 15 objects. For example, assuming a supercomputer can evaluate one million partitions per second, the time required by Jensen's algorithm to optimally partition 25 objects into 10 groups is 16 hr. While at first glance this quantity appears very large, it actually represents a tremendous reduction from the 3,800 years

required by the same computer to solve the problem by complete enumeration. Still, since commonality analyses have been defined which involve 100 or more objects, the computational requirements of Jensen's method are obviously unacceptable.

Anderberg [25] suggests that upon determining the optimum partition via a hierarchical method, a switching algorithm such as the k-means algorithm be employed to improve the solution. Beginning with an initial partition, switching algorithms consider each individual item to determine whether a move to another cluster improves the solution, and repeats this process until no move of a single individual results in further improvement. He observes that since the hierarchical solutions are, in general, very close to optimum the heavy computational burden associated with the switching algorithms is not prohibitive. The steps involved in the use of a switching algorithm are as follows:

- 1) Generate an initial partition or set of N clusters. A hierarchical solution is commonly used as the initial partition.
- 2) Take each object in sequence and evaluate the distance  $D(u,v)$  to each of the N clusters. For item u which is a member of cluster k, if  $D(u,v)$  is minimum for some v unequal to k, then switch object u to cluster v.
- 3) Repeat step (2) until no further switches decrease the objective function.

In general, the composition of clusters can change but the number of clusters will not change from the application of a switching algorithm. While single object moves will not necessarily generate an optimum solution, some improvement may be expected.

Of the many clustering methods available, a variance-based hierarchical clustering algorithm, which incorporates a switching algorithm for optimization, appears to be the most applicable to commonality analysis. This type of clustering algorithm was recently used to assess the commonality of the space station rack utility interfaces [26] and showed significant potential for more general application to commonality analysis. A generalized development of this clustering algorithm is presented in the following section.

#### IV. A CLUSTERING BASED COMMONALITY ANALYSIS METHODOLOGY

##### A. Specification of an Object

In a cluster analysis, the objects to be grouped typically have one or more attributes which manifest the similarity between the objects. For the purpose of commonality analysis, an object or item may be characterized by a real attribute vector. Given a group of items for commonality analysis, the attribute vector is m dimensional, with each attribute representing a function required by at least one item. Using underscores to denote vectors, let the attribute vector for item i be given as

$$\underline{x}_i = \langle x_{i,1}, x_{i,2}, \dots, x_{i,m} \rangle$$

where  $x_{i,j}$  is a nonnegative real number. For each  $x_{i,j}$ , the real entry represents the quantity of a particular function  $j$  possessed by item  $i$ , with  $x_{i,j}$  being zero if that function is not required.

For example, consider the set of tanks given in Table 2, specified during the preliminary design activity of the space station program.

TABLE 2. SPACE STATION WATER TANKS

Tank	Quantity	Weight(lbs)	Volume(ft <sup>3</sup> )	R&D(\$k)	Prod(\$k)
1	2	3.23	16.70	92.82	102.51
2	6	27.87	135.67	366.68	810.76
3	4	1.98	9.61	67.83	64.60
4	2	26.58	129.41	355.77	775.18
5	4	2.16	10.53	71.86	70.60
6	1	1.20	5.83	49.37	40.20
7	1	1.08	5.25	46.17	36.12
8	4	40.36	196.50	464.31	1152.11
9	3	29.26	142.42	378.24	844.66

Each tank  $i$  can be represented by a four-dimensional attribute vector where

$x_{i,1}$  = weight in pounds,

$x_{i,2}$  = volume in cubic feet,

$x_{i,3}$  = research and development cost in thousands of dollars, and

$x_{i,4}$  = production cost in thousands of dollars.

The attribute vector for tank 3 would then be

$$\underline{x}_3 = < 1.975, 9.61, 67.833, 64.598 > .$$

In this example all attributes are nonzero. However, if one of the tanks required a heater interface, the value of the heater interface attribute would be zero for the other tanks.

## B. Specification of a Common Item

The mean attribute vector defines the span of common functions required by the items within the cluster. The mean attribute vector for a cluster  $k$  is given as

$$\underline{\mu}_k = \langle \mu_{k,1}, \mu_{k,2}, \dots, \mu_{k,m} \rangle$$

analogous to the definition of the attribute vector  $\underline{x}_i$ . Consider a function vector  $\underline{\beta}$  which defines the operations by which each component of the mean attribute vector  $\underline{\mu}$  is determined:

$$\underline{\beta} = \langle \beta_1, \beta_2, \dots, \beta_m \rangle .$$

Let each operation  $\beta_j$  in  $\underline{\beta}$  be defined such that

$$x \leq (x \beta_j y) \leq y$$

for two nonnegative real numbers  $x$  and  $y$ , and let the following conditions hold over the set of items  $Q$  in a commonality analysis:

- 1) Identity -  $\underline{x}_u \underline{\beta} \underline{x}_v = \underline{x}_u$  if  $\underline{x}_u = \underline{x}_v$  for all items  $u, v \in Q$ .
- 2) Commutativity -  $\underline{x}_u \underline{\beta} \underline{x}_v = \underline{x}_v \underline{\beta} \underline{x}_u$  for all items  $u, v \in Q$ .
- 3) Associativity -  $\underline{x}_u \underline{\beta} (\underline{x}_v \underline{\beta} \underline{x}_w) = (\underline{x}_u \underline{\beta} \underline{x}_v) \underline{\beta} \underline{x}_w$  for all items  $u, v, w \in Q$ .

For attributes such as structural load, flow rate, and MTBF,  $\beta_j$  will typically constitute a maximum function whereas for attributes such as vibration, noise emission, and error rate,  $\beta_j$  will typically constitute a minimum function.

For two items  $u$  and  $v$ , the mean attribute vector  $\underline{\mu}$  of a common item  $uv$  is given as

$$\underline{\mu}_{uv} = \langle \mu_{uv,1}, \mu_{uv,2}, \dots, \mu_{uv,m} \rangle$$

where each element  $\mu_{uv,i}$  may be determined as

$$\mu_{uv,j} = x_{u,j} \beta_j x_{v,j} \quad , \quad j=1,2,\dots,m$$

or, in vector notation,

$$\underline{\mu}_{uv} = \underline{x}_u \beta \underline{x}_v \quad .$$

In general, the mean attribute vector may be computed as a function of the attributes of all items in the cluster as follows:

$$\begin{aligned} \underline{\mu}_k &= (\dots((\underline{x}_1 \beta \underline{x}_2) \beta \underline{x}_3 \dots \beta \underline{x}_M) \quad , \quad \underline{x}_i \in \text{cluster } k, i = 1, 2, \dots, M \quad (1) \\ &= \langle (\dots(x_{1,1} \beta_1 x_{2,1}) \beta_1 x_{3,1}) \dots \beta_1 x_{M,1}), \\ &\quad (\dots(x_{1,2} \beta_2 x_{2,2}) \beta_2 x_{3,2}) \dots \beta_2 x_{M,2}), \\ &\quad \dots, (\dots(x_{1,m} \beta_m x_{2,m}) \beta_m x_{3,m}) \dots \beta_m x_{M,m}) \rangle \end{aligned}$$

where that are up to M items in cluster k and the attribute vector is m dimensional. Other functions satisfying the conditions for  $\beta$  operations, such as an arithmetic average, may be utilized for  $\beta_j$  as needed to define the components of the mean attribute vector.

For example, consider the development of a common tank to replace tanks 6 and 7 given in Table 2. The attribute vectors for tanks 6 and 7 are

$$\underline{x}_6 = \langle 1.2, 5.83, 49.374, 40.204 \rangle$$

and

$$\underline{x}_7 = \langle 1.08, 5.25, 46.166, 36.116 \rangle \quad .$$

Let a larger capacity tank be a technically feasible replacement for a smaller tank, and define the function vector as

$$\underline{\beta} = \langle \text{Max}(y, z), \text{Max}(y, z), \text{Max}(y, z), \text{Max}(y, z) \rangle$$

where

$$\begin{aligned} \text{Max}(y, z) &= y \text{ if } y \geq z \\ &= z \text{ if } y < z \text{ for two real variables } y \text{ and } z \geq 0 \end{aligned}$$

so that the attributes of the common tank will satisfy the more critical requirements of the larger tank. Then the mean attribute vector for tanks 6 and 7 becomes

$$\underline{\mu}_6 = < 1.2, 5.83, 49.374, 40.204 > .$$

The function  $\text{Max}(y,z)$  is very common in commonality applications and, since all  $\beta_j$  were maximum functions, led to a mean attribute vector equal to the attribute vector for tank 6. This outcome was observed because all attributes of the tanks are co-increasing; an increase in any attribute implies increases in all other attributes. In general this outcome will not occur.

In general the attributes of the common item defined by the mean attribute vector  $\underline{\mu}$  differ from each constituent item of that cluster. These differences represent the excess functionality following from use of the common item in place of the individual items. The excess functionality which results from commonality may then be quantified as the sum of the differences between the mean attribute vector and the individual attribute vectors of the members of a cluster. Let the quantity  $S$  define the excess functionality which results from partitioning  $M$  items into  $N$  groups,  $N \leq M$ , and let  $\sigma$  define the excess functionality present in any one of the  $N$  groups. Then  $S$  is given as

$$S = \sum_k \sigma_k, \quad k = 1, 2, \dots, N \quad (2)$$

for each of  $N$  clusters  $k$ , where

$$\sigma_k = \sum_{i \in k} QN_i \left\{ \sum_{j=1}^m C_j |x_{i,j} - \mu_{k,j}| \right\}, \quad x_i \in \text{cluster } k, \quad i=1, 2, \dots, M \quad (3)$$

gives the excess functionality for each cluster. In equation (3), the constant  $C_j$  defines the relative weighting of the individual attributes and  $QN_i$  is the number of units of item  $i$ .

To illustrate the computation of the quantity  $\sigma_k$ , consider the previous clustering of tanks 6 and 7. From Table 2, the units of  $x_1$  are pounds while the units of  $x_2$  are cubic feet and the units of  $x_3$  and  $x_4$  are in dollars; the coefficients  $C_j$  may be used to normalize these measures. Let the launch cost of 1 lb be \$2,800 and let the worth of 1 ft<sup>3</sup> of space inside a pressurized module be \$10,000. Since the application of commonality is in essence one of substituting a larger tank for a smaller tank in this instance, the net R&D cost will decrease from commonality by eliminating the development effort of the smaller tanks; this cost decrease can be accounted for in the objective function. However, the production cost will increase since it costs

more to produce the larger tanks than the smaller tanks. Then the excess functionality may be normalized in terms of cost (in \$k) by letting  $C_1 = \$2.8/\text{lb}$ ,  $C_2 = \$10.0/\text{ft}^3$ ,  $C_3 = 0$ , and  $C_4 = 1$ . Using equation (3) with  $QN_6 = QN_7 = 1$ ,

$$\begin{aligned}\sigma_6 &= \sum_{i=6,7} [2.8(|x_{i,1} - \mu_{6,1}|) + 10.0(|x_{i,2} - \mu_{6,2}|) + 0(|x_{i,3} - \mu_{6,3}|) \\ &\quad + 1(|x_{i,4} - \mu_{6,4}|)] \\ &= [2.8(|1.2 - 1.2|) + 10.0(|5.83 - 5.83|) + 1(|40.204 - 40.204|)] \\ &\quad + [2.8(|1.08 - 1.2|) + 10.0(|5.25 - 5.83|) + 1(|36.116 - 40.204|)] \\ &= 10.224 \quad .\end{aligned}$$

Thus, the extra weight, extra volume, and increased manufacturing cost which result when tank 6 is used as a common tank for both tanks 6 and 7 may be quantified as a cost of \$10,224. Hence, tanks 6 and 7 should be made common only if the cost savings resulting from commonality exceed \$10,224. This comparison will be addressed subsequently.

### C. Selection of Commonality Alternatives

The standardization which follows from commonality reduces cost, but the excess functionality which results increases cost; hence, commonality should be applied only when the cost reductions from standardization are larger in magnitude than the cost increases from excess functionality. Recall the economic feasibility of commonality alternatives from Section II. Let the cost of a particular partition which groups  $M$  functionally similar items into  $N$  clusters – with an associated level of excess functionality  $S$  – be described by some cost function  $f(n,s)$  where real variables  $n$  and  $s$  equal  $N$  and  $S$ , respectively. Additionally, let  $f(n,s)$  be a strictly increasing function in  $n$ . Then an objective function of the form

$$Z = f(n,s) \tag{4}$$

may be used to evaluate the net change in cost associated with various levels of commonality. The grouping of items which minimizes  $Z$  is the optimum partition.

Consider the formulation of an objective function for the tankage commonality example discussed previously. The cost to produce the tanks may be determined by summing the research and development cost, the production cost, and the cost increases due to extra weight, volume, and increased complexity. Assume that there is no learning curve present. An objective function of the form



$$Z(n,s) = \left\{ \sum_{k=1}^n \mu_{k,3} \right\} + s \quad (5)$$

may then be used to compare the cost effectiveness of different commonality alternatives with associated values of N and S. Note that the function Z is not a cost estimating function; in this case, the manufacturing cost is not included. However, Z does provide an accounting of the costs which are impacted by commonality. Recall that the increase in manufacturing cost due to commonality is included in S. If it is desired to use Z as a cost function, the constant costs may be incorporated.

Recall from Section III that the distance  $D(u,v)$  between two items is defined as the change in the objective function which will result if the two items are merged into a single cluster. Making two items u and v common will decrease N by one and will increase S. In Appendix A it is shown that for two clusters k and l each comprised of one or more items,

$$\sigma_{kl} = \sigma_k + \sigma_l + \delta(k,l) \quad (6)$$

where  $\delta(k,l) \geq 0$ . Thus, from equations (2) and (6),  $\delta(u,v)$  defines the increase in S which results from making items u and v common.

When a clustering of item u with item v decreases Z, the benefits of commonality outweigh the costs of excess functionality. The cost effectiveness of making items u and v common is then

$$\begin{aligned} D(u,v) &= Z[N-1, S+\delta(u,v)] - Z(N,S) \\ &= f[N-1, S+\delta(u,v)] - f(N,S) \end{aligned} \quad (7)$$

where N and S are the number of clusters and excess functionality for two types of items u and v.  $D(u,v)$  then represents the signed magnitude of the cost impact resulting from making items u and v common.

Again consider the use of a common tank instead of tanks 6 and 7. Using the objective function developed in the previous example, the cost to implement all nine (N=9) types of tanks listed in Table 2 may be determined as

$$\begin{aligned} Z(9,0) &= \left\{ \sum_k \mu_{k,3} \right\} + S \\ &= \{92.819 + 366.685 + \dots + 378.24\} + 0 \\ &\approx 1,893 \end{aligned}$$

where S is zero since there is no excess functionality present. If a common tank is used for tanks 6 and 7, then there are eight types of tanks (N=8) and the excess

functionality ( $S=10.224$ ) was determined in a previous example. The cost to implement these eight types of tanks is then

$$\begin{aligned}
 Z(8, 10.224) &= \left\{ \sum_k \mu_{k,3} \right\} + S \\
 &= \{92.819 + 366.685 + \dots + 49.374 \text{ (common tank)} \\
 &\quad + 464.31 + 378.24\} + 10.224 \\
 &\approx 1,857 \quad .
 \end{aligned}$$

The cost effectiveness of making tanks 6 and 7 common is then

$$\begin{aligned}
 D(6,7) &= Z(8, 6.66) - Z(9, 0) \\
 &= 1,857 - 1.893 \\
 &= -36
 \end{aligned}$$

which represents a net cost savings of approximately \$36,000 from commonality. In this instance, the cost savings resulting from commonality outweigh the cost incurred due to excess functionality, and the commonality application is cost effective.

#### D. Optimality Considerations in the Selection of Commonality Alternatives

Consider the sorting of three objects a, b, and c, into two groups. Given an objective function Z, whose value on the initial partition  $\{[a],[b],[c]\}$  is denoted  $Z_{init}$ , the clustering algorithm previously discussed would cluster these objects by determining the minimum of  $D(a,b)$ ,  $D(a,c)$ , and  $D(b,c)$ . The two objects having the minimum  $D(u,v)$  would then be combined into one group. Let  $D(a,b)$  be the minimum distance; then, objects a and b are clustered to give the partition  $\{[a,b],[c]\}$ . The value of the objective function for this grouping is  $Z_{init} + D(a,b)$ . Since  $D(a,b) \leq D(a,c)$  and  $D(a,b) \leq D(b,c)$ , the objective function is minimized for this set of three objects grouped into two partitions.

Next, consider the hierarchical sorting of four objects a, b, c, and d, into two groups. Let the objective function Z assume the value  $Z_{init}$  on the initial partition  $\{[a],[b],[c],[d]\}$ . As before, let  $\delta(a,b)$  be the minimum distance such that the partition  $\{[a,b],[c],[d]\}$  gives the minimum value of Z for three groups and equals the sum of  $Z_{init}$  and  $D(a,b)$ . One more clustering step must be performed to achieve two groups. A hierarchical clustering method treats the group [a,b] as a single item in that only the distances  $D(c,ab)$ ,  $D(d,ab)$ , and  $D(c,d)$  are evaluated. Again the

minimum  $D(u,v)$  is selected to yield a partition of two groups. With no loss of generality, let  $D(c,ab)$  be the minimum, resulting in the partition  $\{[a,b,c],[d]\}$ .

This partition does not necessarily yield the minimum value of  $Z$  for two groups. Although  $D(a,b) \leq D(b,c)$ , it is possible that  $D(a,b) + D(c,ab) > D(b,c) + D(d,bc)$  or other possible combinations. Still, a sufficient condition for the partition of four items into two groups to yield a minimum value of  $Z$  may be developed.

#### E. S-Minimal Partitions

Recall from Section II that the application of commonality to a given set of items  $u$  and  $v$  decreases some costs and increases others. The cost increases which result from commonality are included in the determination of the quantity of excess functionality  $S$ . If the cost reductions attributable to commonality are constant for the formation of a common item from any two types of items, then the cost effectiveness of commonality is determined solely by the associated excess functionality  $S$ . At any step in the hierarchical clustering procedure, the minimum distance  $D(u,v)$  will occur for those two items having the minimum  $\delta(u,v)$ . In this case, the objective function  $Z$  is then strictly increasing in  $s$  for  $n=N$  and thus may be minimized for a given  $N$  by determining the partition having the minimum  $S$ .

In Appendix A, it is shown that the excess functionality which results from the merger of two clusters is at least as large as the maximum  $\delta(u,v)$  for any two members of the respective clusters:

$$\sigma_k + \sigma_l + \delta(k,l) \geq \text{Maximum } \{\delta(u,v)\} \quad (8)$$

for all items  $u,v$  which are members of either cluster  $k$  or cluster  $l$ . From equation (8), if for two clusters  $k$  and  $l$  each consisting of one or more items,

$$\sigma_k + \sigma_l \leq \text{Minimum } \{\delta(u,v)\} \quad \text{for all } u \in \text{cluster } k \text{ and } v \in \text{cluster } l \quad (9)$$

then any partition including items from both clusters  $k$  and  $l$  will, by equation (8), have excess functionality at least as large as  $\sigma_k + \sigma_l$ . For a given  $N$ , a partition in which all clusters taken pairwise satisfy equation (9) is S-minimal in that no other partition exists which gives a smaller value of  $S$ . If the cost savings of commonality is constant for the clustering of any two items, then this partition is also Z-minimal in that no other partition will further decrease the objective function.

To illustrate the use of the sufficiency condition of equation (9) for S-minimal partitions, consider the partition  $\{(abc),(def),(ghij)\}$  of ten objects into three groups. For this partition, by equation (2)

$$S = \sigma_{abc} + \sigma_{def} + \sigma_{ghij} \quad .$$

Using equation (9), the following inequalities are formed from the pairwise combinations of the groups:

$$\sigma_{abc} + \sigma_{def} \leq \text{Minimum } \{\delta(a,d), \delta(a,e), \dots, \delta(c,f)\} ,$$

$$\sigma_{abc} + \sigma_{ghij} \leq \text{Minimum } \{\delta(a,g), \delta(a,h), \dots, \delta(c,j)\} ,$$

$$\sigma_{def} + \sigma_{ghij} \leq \text{Minimum } \{\delta(d,g), \delta(d,h), \dots, \delta(f,j)\} .$$

If these inequalities hold for the partition  $\{(abc),(def),(ghij)\}$  then it is an S-minimal partition. A proof is given in Appendix A.

#### F. Z-Minimal Partitions

The objective function, as given in equation (4), is a function of real variables  $n$  and  $s$  where  $n$  is the number of clusters and  $s$  is excess functionality. If it is possible to dichotomize the cost impacts of commonality such that

$$Z = f_1(n,s) + f_2(n,s) + C$$

where

$f_1(n,s)$  quantifies the cost savings attributable to commonality,

$f_2(n,s)$  quantifies the cost increases attributable to commonality, and

$C$  quantifies costs not impacted by commonality,

Then in general

$$Z = f_1(n,s) + s + C \tag{10}$$

since the cost increases which result from commonality are included in the determination of the excess functionality  $S$ . Recall that  $S$  is always nonnegative; since  $f_1(n,s)$  only quantifies cost savings, it is always nonpositive.

In the previous section,  $f_1(n,s)$  was treated as a linear function of  $n$  and a sufficient condition for Z-minimal partitions was defined. If  $f_1(n,s)$  is not a linear function of  $n$ , the condition of equation (9) cannot be employed. However, it is possible to bound the changes in  $f_1(n,s)$  which result from the clustering of any two items. This bound, taken in conjunction with equation (9), may be used to verify Z-minimal partitions in general.

Just as the quantity  $S$  denotes the total excess functionality resulting from the use of  $N$  types of items in  $M$  unique applications,  $f_1(N,S)$  denotes the total cost savings which result from this commonality. The quantity  $\sigma_k$  is employed to denote the quantity of excess functionality associated with any one of the  $N$  types of items  $k$ . Let the quantity  $\alpha_k$  likewise denote the cost savings associated with the  $k$ th of  $N$  types of items. The definition of commonality cost savings as

$$f_1(n,s) = \sum_k \alpha_k, \quad \text{for } k = 1, 2, \dots, n \quad (11)$$

is analogous to the definition of excess functionality  $S$  in equation (2).

The quantity  $\alpha_k$  is the sum of all research and development efforts eliminated and the manufacturing cost reductions through the formation of cluster  $k$  or the development of common item  $k$ . If cluster  $k$  consists of only one type of item, then  $\alpha_k$ , like  $\sigma_k$ , is zero. Let the R&D cost for item  $u$  be denoted  $R\&D(u)$  and let the manufacturing cost savings which result from the quantity of item  $u$  be denoted  $MAN(u, QN_u)$  where  $QN_u$  is the quantity of  $u$ . In Appendix A, it is shown that for two clusters  $k$  and  $l$  each composed of two or more items,

$$\begin{aligned} \alpha^*(k) = & [\text{Maximum}\{R\&D(u)\} + \text{Maximum}\{MAN(v, QN_k + QN_l)\}] \\ & - [\text{Minimum}\{R\&D(w)\} + \text{Minimum}\{MAN(y, QN_y)\}] \end{aligned} \quad (12)$$

for all  $u, v, w, y \in$  cluster  $k$ , is the maximum reduction in  $f_1(N,S)$  which can be incurred from switching one or more items from cluster  $k$  to cluster  $l$ . If cluster  $k$  contains only one item,  $\alpha^*(k)$  is undefined since merging it with another cluster would reduce  $N$ .

If inequality (9) holds for two clusters  $k$  and  $l$  and the minimum  $\delta(u,v)$  is sufficiently large, then this  $S$ -minimal partition will also be  $Z$ -minimal. Given two clusters  $k$  and  $l$  each comprised of one or more items, if

$$\sigma_k + \sigma_l + \text{Maximum}\{\alpha^*(k), \alpha^*(l)\} \leq \text{Minimum}\{\delta(u,v)\} \quad (13)$$

for all  $u \in$  cluster  $k$  and  $v \in$  cluster  $l$ , then any partition including items of cluster  $k$  and cluster  $l$  cannot reduce the objective function  $Z$ . The term

$$\text{Maximum}\{\alpha^*(k), \alpha^*(l)\}$$

is used to make the condition binding for switching items from cluster  $k$  to cluster  $l$  or for switching items from cluster  $l$  to cluster  $k$ . Equation (13) defines a sufficient condition for a  $Z$ -minimal partition of  $M$  items into  $N$  groups when an objective

function of the form given in equation (10) may be used. For a given N, if all clusters taken pairwise satisfy equation (13), the partition is Z-minimal.

To illustrate the use of this condition, consider the sorting of tanks 2, 4, 6, and 7 from Table 2 into two groups. The objective function employed in a prior example [equation (5)] is of the form given in equation (10), and it will be employed here also. From the prior example, it was determined that  $D(6,7) = -36$ . In like fashion, it may be determined that  $D(2,4) = -150$ . The partition  $\{(2,4),(6,7)\}$  then reduces the objective function by a total of -186. The excess functionality  $\sigma_{6,7}$  between tanks 6 and 7 was also determined in an earlier example. Using the same method, the excess functionality which results from any pairwise combination of these four tanks is tabulated in the following matrix (in \$K):

	2	4	6	7
2	-	-	-	-
4	203	-	-	-
6	2,144	2,042	-	-
7	2,154	2,052	10	-

Denoting the two groups comprising the partition as clusters 2,4 and 6,7, it is observed that  $\sigma_{2,4} = 203$  and  $\sigma_{6,7} = 10$ . It is also observed that the minimum inter-cluster  $\delta(u,v)$  is  $\delta(4,6) = 2,042$ . By the condition of equation (9), the partition  $\{(2,4),(6,7)\}$  is S-minimal since

$$\sigma_{2,4} + \sigma_{6,7} \leq \text{Minimum}\{\delta(u,v)\} \quad \text{for } u = 2,4 \quad \text{and } v = 6,7$$

$$213 \leq 2,042 \quad .$$

The maximum cost savings which could result from switching either tank 2 or tank 4 to cluster 6,7 is

$$\begin{aligned} \alpha^*(2,4) &= \text{Maximum}\{x_{2,3}, x_{4,3}\} - \text{Minimum}\{x_{2,3}, x_{4,3}\} \\ &= \text{Maximum}\{366, 355\} - \text{Minimum}\{366, 355\} \\ &= 11 \end{aligned}$$

by equation (12). Likewise, the maximum cost savings which could result from switching either tank 6 or 7 to cluster 2,4 is

$$\alpha^*(6,7) = 3$$

also by equation (12). Now,

$$\begin{aligned}\sigma_{2,4} + \sigma_{6,7} + \text{Maximum}\{\alpha^*(2,4), \alpha^*(6,7)\} &= 203 + 10 + 11 \\ &= 224 \quad .\end{aligned}$$

Since the minimum intercluster  $\delta(u,v) = 2,042$ , then by the sufficient condition of equation (13) the partition  $\{(2,4), (6,7)\}$  is Z-minimal; none of the other six ways of sorting these four types of tanks into two groups can yield a lower value of the objective function.

By varying the number of clusters  $N$  and determining the Z-minimal partition for each value of  $N$ , the optimal partition and globally minimum value of the objective function  $Z$  can be established. However, the conditions of equations (9) and (13) are only sufficient for a Z-minimal partition; neither is a necessary condition. Therefore, even though a partition may indeed be Z-minimal, it may not be possible to verify it without complete enumeration. In these cases, the sufficiency conditions may be employed in a branch-and-bound algorithm similar to that of Yeager [15] to eliminate some partitions from the solution space.

#### G. Illustration of Methodology

To demonstrate this commonality analysis methodology, consider the implementation of the clustering algorithm discussed in Section III using the tank data of Table 2. The first part of the clustering algorithm employs hierarchical clustering techniques to approximate an optimal commonality solution. The specification of an object and the objective function to be minimized were discussed in prior examples and will be used here without introduction.

Step 1. The algorithm begins with no commonality assumed, or  $N$  clusters of one type of object each. In this case, the partition  $\{1,2,3,4,5,6,7,8,9\}$  represents the "no commonality" option.

Step 2. Equation (7) may be employed to calculate the distance  $D(u,v)$  between each type of tank using the objective function given in equation (5). These distances comprise the entries in the similarity matrix given in Table 3.

Step 3. Note from Table 3 that the minimum entry is  $D(2,4) = -0.152$ , indicating that use of a common tank instead of tanks 2 and 4 is the most cost effective commonality alternative.

Step 4. Objects 2 and 4 are clustered and treated as a single object, object 2, in subsequent steps. Object 4 is removed from the similarity matrix.

Step 5. Steps 2 through 4 are repeated until no mergers are possible which further reduce the objective function, or all entries in the similarity matrix are positive. Step 2 yields the similarity matrix given in Table 4. In step 3, inspection of Table 4 reveals that  $D(6,7) = -0.036$  is the minimum entry, and thus these two items are clustered in step 4. Continuing in this fashion, tanks 3 and 5 are clustered in the following cycle. After clustering tanks 3 and 5, the next similarity matrix, given in Table 5, contains no negative entries, indicating that no further commonality is cost effective.

TABLE 3. INITIAL SIMILARITY MATRIX FOR SPACE STATION  
WATER TANKS (\$M)

	1	2	3	4	5	6	7	8	9
1	-	-	-	-	-	-	-	-	-
2	3.8	-	-	-	-	-	-	-	-
3	0.38	8.2	-	-	-	-	-	-	-
4	3.6	-0.152	7.8	-	-	-	-	-	-
5	0.31	8.2	-0.005	7.8	-	-	-	-	-
6	0.13	2.1	0.015	2.0	0.031	-	-	-	-
7	0.14	2.1	0.028	2.0	0.044	-0.036	-	-	-
8	5.8	5.5	12.2	1.8	12.1	3.1	3.1	-	-
9	4.1	0.27	8.7	0.058	8.6	2.2	2.2	2.3	-



TABLE 4. SECOND SIMILARITY MATRIX FOR SPACE STATION  
WATER TANKS (\$M)

	1	2	3	5	6	7	8	9
1	-	-	-	-	-	-	-	-
2	3.8	-	-	-	-	-	-	-
3	0.38	8.2	-	-	-	-	-	-
5	0.31	8.2	-0.005	-	-	-	-	-
6	0.13	2.1	0.015	0.031	-	-	-	-
7	0.14	2.1	0.028	0.044	-0.036	-	-	-
8	5.8	7.5	12.2	12.1	3.1	3.1	-	-
9	4.1	0.48	8.7	8.6	2.2	2.2	2.3	-

TABLE 5. FINAL SIMILARITY MATRIX FOR SPACE STATION  
WATER TANKS (\$M)

	1	2	3	6	8	9
1	-	-	-	-	-	-
2	3.8	-	-	-	-	-
3	0.70	16.4	-	-	-	-
6	0.30	4.2	0.11	-	-	-
8	5.8	7.5	24.3	6.2	-	-
9	4.1	0.48	17.3	4.4	2.3	-

The hierarchical approximation to the optimal commonality solution is  $\{1, (2,4), (3,5), (6,7), 8, 9\}$  and the nine original types of tanks have been replaced by six types of tanks. The cost of the excess functionality resulting from this partition is approximately \$272,000. Recall that  $Z(9,0)=1,893$  (in \$k) from a previous example. The value of the objective function for this partition  $[Z(6,272)]$  is 1,700 (in \$k), representing a total cost reduction of approximately \$193,000.

The second part of the clustering algorithm employs switching methods to improve the hierarchical solution. The definition of an object and the objective function are unchanged from those used above.

Step 1. The hierarchical approximation determined above will be used as the initial partition:  $\{1, (2,4), (3,5), (6,7), 8, 9\}$ .

Step 2. Equation (7) may then be employed to determine the distances between the individual objects and the other clusters, for example the distance between object 4 and cluster  $(6,7)$ . The switching similarity matrix which results is given in Table 6.

Step 3. Inspection of Table 6 reveals that in all cases the minimum distance for each object is for the cluster in which the object is a member. Thus, any switch would increase the objective function and no further steps are required.

Therefore, the clustering solution to this tankage commonality analysis problem is defined by the partition  $\{1, (2,4), (3,5), (6,7), 8, 9\}$ .

TABLE 6. MODIFIED SIMILARITY MATRIX FOR SPACE STATION  
WATER TANKS (\$M)

	1 <sup>*</sup>	(2,4) <sup>*</sup>	(3,5) <sup>*</sup>	(6,7) <sup>*</sup>	8 <sup>*</sup>	9 <sup>*</sup>
1	-	3.8	0.70	0.30	5.8	4.1
2	3.8	-0.15	16.3	4.16	5.5	0.27
3	0.38	8.2	-0.005	0.055	12.2	8.7
4	3.6	-0.15	15.6	4.0	1.8	0.058
5	0.31	8.2	-0.005	0.082	12.1	8.6
6	0.13	2.1	0.023	-0.036	3.1	2.2
7	0.14	2.1	0.040	-0.036	3.1	2.2
8	5.8	7.5	24.3	6.2	-	2.3
9	4.1	0.48	17.3	4.4	2.3	-

Since the switching portion of the algorithm did not change the hierarchical approximation and because the size of the data set is relatively small ( $M=9$ ), this approximate solution may indeed be the optimum. The sufficient conditions for Z-minimal partitions developed earlier may be employed to examine the optimality of this solution. The increases in  $S$ ,  $\delta(u,v)$ , which result from the clustering of any two tanks is given in Table 7. The maximum increase in cost savings,  $\alpha^*$ , possible from switching tanks between clusters is given in Table 8. The excess functionality  $\sigma_k$  associated with each cluster is given in Table 9. The partition  $\{1, (2,4), (3,5), (6,7), 8, 9\}$  is comprised of six clusters – three single item clusters and three double item clusters. Since the sufficiency condition of equation (13) applies to clusters taken pairwise, a total of 15 inequalities must be examined. These inequalities are as follows:

1.  $\sigma_1 + \sigma_2 + \text{Maximum}\{\alpha^*(1), \alpha^*(2)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 1, v = 2, 4$
2.  $\sigma_1 + \sigma_3 + \text{Maximum}\{\alpha^*(1), \alpha^*(3)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 1, v = 3, 5$
3.  $\sigma_1 + \sigma_6 + \text{Maximum}\{\alpha^*(1), \alpha^*(6)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 1, v = 6, 7$
4.  $\sigma_1 + \sigma_8 + \text{Maximum}\{\alpha^*(1), \alpha^*(8)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 1, v = 8$
5.  $\sigma_1 + \sigma_9 + \text{Maximum}\{\alpha^*(1), \alpha^*(9)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 1, v = 9$
6.  $\sigma_2 + \sigma_3 + \text{Maximum}\{\alpha^*(2), \alpha^*(3)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 2, 4, v = 3, 5$
7.  $\sigma_2 + \sigma_6 + \text{Maximum}\{\alpha^*(2), \alpha^*(6)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 2, 4, v = 6, 7$
8.  $\sigma_2 + \sigma_8 + \text{Maximum}\{\alpha^*(2), \alpha^*(8)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 2, 4, v = 8$
9.  $\sigma_2 + \sigma_9 + \text{Maximum}\{\alpha^*(2), \alpha^*(9)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 2, 4, v = 9$
10.  $\sigma_3 + \sigma_6 + \text{Maximum}\{\alpha^*(3), \alpha^*(6)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 3, 6, v = 6, 7$
11.  $\sigma_3 + \sigma_8 + \text{Maximum}\{\alpha^*(3), \alpha^*(8)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 3, 6, v = 8$
12.  $\sigma_3 + \sigma_9 + \text{Maximum}\{\alpha^*(3), \alpha^*(9)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 3, 6, v = 9$
13.  $\sigma_6 + \sigma_8 + \text{Maximum}\{\alpha^*(6), \alpha^*(8)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 6, 7, v = 8$
14.  $\sigma_6 + \sigma_9 + \text{Maximum}\{\alpha^*(6), \alpha^*(9)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 6, 7, v = 9$
15.  $\sigma_8 + \sigma_9 + \text{Maximum}\{\alpha^*(8), \alpha^*(9)\} \leq \text{Minimum}\{\delta(u,v)\}$  ,  $u = 8, v = 9$  .

From Tables 7, 8, and 9, it may be determined that all inequalities hold except for inequality number 10. Thus, in order to verify the partition as Z-minimal, some enumeration of alternatives will be required.

TABLE 7. INCREASE IN EXCESS FUNCTIONALITY [ $\delta(u,v)$ ] FROM THE CLUSTERING OF ANY TWO TANKS (\$M)

	1	2	3	4	5	6	7	8	9
1	-	-	-	-	-	-	-	-	-
2	3.93	-	-	-	-	-	-	-	-
3	0.45	8.32	-	-	-	-	-	-	-
4	3.73	0.20	7.91	-	-	-	-	-	-
5	0.39	8.25	0.062	7.85	-	-	-	-	-
6	0.18	2.14	0.064	2.04	0.08	-	-	-	-
7	0.19	2.15	0.07	2.05	0.09	0.01	-	-	-
8	5.90	5.91	12.26	2.17	12.19	3.13	3.14	-	-
9	4.14	0.63	8.74	0.41	8.68	2.25	2.26	2.64	-

TABLE 8. MAXIMUM COST SAVINGS FROM SWITCHING TANKS BETWEEN CLUSTERS

Cluster (k)	Members	$\alpha_k^*$ (\$M)
1	1	-
2	2,4	0.011
3	3,5	0.004
6	6,7	0.003
8	8	-
9	9	-

TABLE 9. EXCESS FUNCTIONALITY OF CLUSTERS IN CLUSTERING SOLUTION

Cluster (k)	Members	$\sigma_k$ (\$M)
1	1	0
2	2,4	0.203
3	3,5	0.063
6	6,7	0.010
8	8	0
9	9	0

Since only inequality number 10 is violated, the only possible partitions which could reduce the objective function involve switching tanks between clusters 3 and 6. The given partition of tanks 3, 5, 6, and 7 is  $\{(3,5),(6,7)\}$ . Other partitions of these four tanks into two groups include

$$\begin{array}{ll} \{3, (5, 6, 7)\} & \{(3, 6), (5, 7)\} \\ \{(3, 5, 6), 7\} & \{(3, 7), (5, 6)\} \\ \{(3, 5, 7), 6\} & \{5, (3, 6, 7)\} \end{array}$$

and it may be verified that each of these partitions results in a larger value of the objective function  $Z$  than the partition  $\{(3,5),(6,7)\}$ . Therefore, using a combined approach of the sufficiency condition and enumeration in the manner of a branch-and-bound algorithm, the partition  $\{1, (2, 4), (3, 5), (6, 7), 8, 9\}$  is verified as a  $Z$ -minimal partition of nine objects into six groups. In this case, the clustering solution was a  $Z$ -minimal partition.

With regard to the optimality of the partition  $\{1, (2, 4), (3, 5), (6, 7), 8, 9\}$ , recall that  $Z$ -minimal partitions must be generated for all  $N$ . The  $Z$ -minimal partitions were determined for  $N = 1, 2, \dots, 9$  by varying the level of  $D(u, v)$  which terminates the hierarchical clustering procedure. In all cases, the clustering solution was a  $Z$ -minimal partition. These partitions and resulting values of the objective function are given in Table 10. Using the data of Tables 7, 8, and 9 together with the fact that  $\sigma_2 = 0.85$  for the group  $(2, 4, 9)$ ,  $\sigma_2 = 9.67$  for the group  $(2, 4, 8, 9)$ ,  $\sigma_3 = 0.60$  for the group  $(3, 5, 6, 7)$  and,  $\sigma_3 = 1.61$  for the group  $(1, 3, 5, 6, 7)$ , the partitions of Table 10 may be verified as  $Z$ -minimal. (Some enumeration is required to verify the partition for  $N=2$  as  $Z$ -minimal.)

TABLE 10.  $Z$ -MINIMAL PARTITIONS OVER RANGE OF  $N$

$N$	$Z$ -Minimal Partition	$Z(N, S)$
1	$\{(1, 2, 3, 4, 5, 6, 7, 8, 9)\}$	47,800
2	$\{(1, 3, 5, 6, 7), (2, 4, 8, 9)\}$	12,474
3	$\{(1, 3, 5, 6, 7), (2, 4, 9), 8\}$	3,180
4	$\{1, (2, 4, 9), (3, 5, 6, 7), 8\}$	2,286
5	$\{1, (2, 4), (3, 5, 6, 7), 8, 9\}$	1,810
6	$\{1, (2, 4), (3, 5), (6, 7), 8, 9\}$	1,700
7	$\{1, (2, 4), 3, 5, (6, 7), 8, 9\}$	1,705
8	$\{1, (2, 4), 3, 5, 6, 7, 8, 9\}$	1,741
9	$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$	1,893

Note from Table 10 that the minimum value of the objective function is achieved for  $N=6$ , and thus the optimal partition of the nine tanks in Table 2 is  $\{1,(2,4),(3,5),(6,7),8,9\}$  and the minimum value of the objective function is  $Z = 1,700$ . As commonality is increased beyond  $N=6$  to  $N=5$ , the costs incurred due to excess functionality are larger than the cost savings due to commonality, and the value of the objective function increases to 1,810. In fact, the costs incurred due to excess functionality for further increases in commonality ( $N=4,3,2,1$ ) are so large that these options actually cost more than the "no commonality" option ( $N=9$ ).

## V. COMMONALITY ANALYSIS AND THE SYSTEM DEVELOPMENT CYCLE

### A. General

The definition of a commonality analysis methodology, as described in the preceding sections, is a significant step. However, knowing how to solve a problem does not give answers unless one knows what problems to solve. Since commonality analysis is an emerging systems-engineering technique, little is known about how to apply commonality analysis in the evolutionary development of an aerospace system. In effect, now that the "how" of commonality analysis has been defined, the "where" must be addressed.

### B. The System Design Process

The aerospace systems engineering process is typically divided into four steps or project phases:

Phase A: Conceptual Design,

Phase B: Preliminary Design,

Phase C: Detail Design,

Phase D: Development.

The major resource expenditure for system development occurs in phases C and D. The primary objective of phase A and B studies is to provide a technical, scientific, and programmatic basis for the commitment of major resources to phases C and D of a project [27].

The initial step in Conceptual Design is a needs analysis. The purpose of the needs analysis is to identify the functional requirements of the system. A feasibility analysis is then conducted to identify alternative candidate systems and compare their strengths and weaknesses. Ostrofsky [28] describes the feasibility analysis as the sorting of the system functions, identified in the needs analysis, into groups which tend to be independent. These groups comprise the subsystems, which are in turn subdivided into major components and so on until a candidate system is defined. For a large scale system, there may be many alternate groupings of the system functions, and thus many candidate systems. By the conclusion of Conceptual Design, an initial screening of candidate system configurations has been performed to identify those which best satisfy the functional requirements of the system.

The Preliminary Design phase adds detail to the candidate systems developed in Conceptual Design and selects an optimal system specification. Feasibility studies are conducted to determine the optimal subsystem and component compositions for each candidate system. Using this increased quantity and quality of information as a basis, system performance models are constructed and used to compare the candidate systems. This system evaluation effort will lead to the selection of an optimal approach – that candidate system configuration which best satisfies the functional requirements of the system identified in the Conceptual Design phase. This system specification is then studied to further define the intersubsystem and intrasubsystem interfaces, associated functional allocations, and performance requirements to the subsystems and components. By the conclusion of Preliminary Design, a single definitive baseline configuration has been specified.

In the prior design phases, activity was directed toward an efficient choice of the optimal candidate system. The Detail Design and Development phases are oriented toward implementation of the optimal system configuration and are frequently referenced together as Phase C/D. The Detail Design activity defines manufacturing and testing plans for all of the parts of the system, develops schedules, and estimates cost. Prototype parts are typically constructed to verify the engineering approach and conformance to system requirements. The Detail Design phase culminates in design specifications for the various parts of the system. The Development phase then constructs the parts in accordance with the detailed system specification. A test program is used in the Development phase to qualify the item in terms of operational requirements.

Design reviews are employed as milestones during the Conceptual, Preliminary, and Detail Design phases of the system design process. The objective of a design review is to identify deficiencies in the design and to recommend corrective actions. Deficiencies are not necessarily flaws in the design, but can involve areas such as conflicting system requirements or cost considerations. Although the exact nature of the reviews may vary, most programs include four basic reviews. They include the Conceptual Design Review near the conclusion of Conceptual Design, the System Design Review near the conclusion of Preliminary Design, and the Equipment and Critical Design Reviews during the Detail Design phase [1]. The Conceptual and Systems Design Reviews cover the system evolution in the Conceptual and Preliminary Design phases, respectively. The Equipment Design Review is scheduled near the beginning of the engineering release of detailed system specifications for manufacturing and establishes baseline design requirements. The Critical Design Review is scheduled near the completion of the engineering release of detailed system specifications and establishes the design baseline.

### C. The Role of Commonality Analysis in Conceptual Design

The objective of commonality analysis is to determine which, if any, items in a set should be developed as a single item and which should not. In Sections II and IV it was shown that the cost effectiveness of commonality is a function of the excess functionality which results from making two or more items common. As a result, items having relatively distinct functions will not, in general, be cost effective if developed as a common item.

One principal activity during Conceptual Design is the definition of candidate system configurations. The various candidate systems are formed by different subdivisions of the system into subsystems. The subsystems are by definition largely

independent and, thus, have distinct functions. For this reason, in general, there exists little potential for commonality at the subsystem level. As the major components of the respective subsystems are specified and, in turn, themselves subdivided, the opportunity for functional similarity between items both within and between subsystems increases. For example, electric motors may be required to power blowers in the air conditioning subsystem and to power pumps in the thermal control system. However, in general, this level of detail is not present in Conceptual Design, restricting the application of commonality analysis.

In some cases, commonality may be analyzed at a high enough level to be of use in Conceptual Design. The common cockpit for the Boeing 757 and 767 aircraft constitutes one example. While the common cockpit was not studied until the latter design phases for the aircraft, Boeing observed that earlier recognition of the advantages of a common cockpit would have increased its cost effectiveness:

[The 757/767 common cockpit] ... resulted in additional weight for the 757 [airplane]. The increased weight, however, was determined to be cost effective when evaluated against the advantages gained by the common type (pilot) rating, as well as common hardware for the two airplanes... If the cockpit commonality goal had been established earlier during the concept design phase of the two airplanes, the weight penalty for the 757 [airplane] may not have been as great [3].

In summary, the role of commonality analysis in Conceptual Design is generally restricted to high level applications such as a common subsystem between two different systems. In these instances where commonality may be explored, recommendations for and against commonality should be addressed in the Conceptual Design Review for the various candidate systems [29]. If commonality analysis is overlooked, the full cost effectiveness of its application may not be realized. While the zealous pursuit of commonality to low levels during Conceptual Design is not warranted, commonality analysis must not be deferred to later design phases.

#### D. Commonality Analysis of Space Station Module Berthing Interfaces

A study performed during the Conceptual Design phase of the space station involved the berthing interface between space station modules [30]. The manner in which utilities must be passed between modules was identified as a key area affecting the cost and feasibility of one particular candidate configuration of the space station. The study was performed to determine the feasibility and composition of a standard interface between modules. Several heuristic strategies were employed in this commonality analysis and recommendations for a common berthing interface were offered as a result.

The berthing interfaces are required to perform various subsets of 25 functions. Thus, a 25-dimensional attribute vector is required to define an interface as follows:

$$\underline{x}_i = \langle x_{i,1}, x_{i,2}, \dots, x_{i,25} \rangle \quad .$$

A listing of these functions is given in Table 11. A total of 48 berthing interfaces were identified for this study. A matrix defining the attribute vector of each interface is given in Table 12. The attribute vector for interface number 25 is



TABLE 11. BERTHING INTERFACE FUNCTIONS

Number	Function
1	Power
2	Thermal Fluid
3	Commands Data
4	Communications
5	Attached Load
6	Attached Torque
7	Water
8	Oxygen and Nitrogen Supply
9	Return Air
10	Active Module Berth
11	Passive Module Berth
12	Active Shuttle Dock
13	Passive Shuttle Dock
14	Active Pallet Dock
15	Passive Pallet Dock
16	Active TMS Berth
17	Passive TMS Berth
18	IVA
19	EVA Airlock
20	Rotary Joint
21	Active Satellite Servicing
22	Passive Satellite Servicing
23	Remote Manipulator System
24	EVA Workstation
25	Remote Manipulator System Grapple

TABLE 12. MODULE BERTHING INTERFACES

Number	Attribute Vector
1	< 1,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0 >
2	< 1,1,1,1,1,1,0,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
3	< 1,1,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 >
4	< 1,1,1,1,1,1,0,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
5	< 1,1,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
6	< 1,1,1,1,1,1,0,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
7	< 1,1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
8	< 1,0,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0 >
9	< 1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0 >
10	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 >
11	< 1,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
12	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
13	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 >
14	< 1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
15	< 1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
16	< 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 >
17	< 1,1,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
18	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 >
19	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0 >
20	< 1,1,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
21	< 1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
22	< 1,0,1,0,1,1,1,1,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >

TABLE 12. (Concluded)

Number	Attribute Vector
23	< 0,0,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0 >
24	< 0,0 >
25	< 1,0,1,0,1,1,1,1,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
26	< 1,1,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
27	< 1,0,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
28	< 1,0,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
29	< 1,1,1,1,1,1,0,1,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
30	< 1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
31	< 1,1,1,1,1,1,1,1,1,1,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0 >
32	< 1,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
33	< 0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
34	< 1,1,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0 >
35	< 0,1,0,0,0,0 >
36	< 1,0,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
37	< 1,0,1,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
38	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0,0 >
39	< 1,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 >
40	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,1,0 >
41	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 >
42	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 >
43	< 1,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0 >
44	< 0,0,1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0 >
45	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1 >
46	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1 >
47	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,1 >
48	< 1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0 >

$$\underline{x}_{25} = \langle 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 \rangle$$

where an entry of 1 signifies that function is required at that interface and an entry of 0 signifies that function is not required at that interface.

A common interface must provide the functions required of each of the unique interfaces which it is to replace. Thus, if a function is required by any interface in a cluster, it must be furnished to all interfaces within that cluster. The  $\underline{\beta}$  function vector is then composed of maximum functions:

$$\begin{aligned}\underline{\beta} &= \langle \beta_1, \beta_2, \dots, \beta_{25} \rangle \\ &= \langle \text{Max}(u,v), \text{Max}(u,v), \dots, \text{Max}(u,v) \rangle\end{aligned}$$

where for two real numbers  $u$  and  $v$ ,

$$\begin{aligned}\text{Max}(u,v) &= u \text{ if } u \geq v \\ &= v \text{ otherwise} \quad .\end{aligned}$$

The mean attribute vector for a common interface to replace interfaces number 25 and 26 is then

$$\underline{\mu}_{25} = \langle 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0 \rangle$$

by equation (1). Since all interfaces are specified in a quantity of one, by equation (3) the excess functionality  $\sigma$  for a common berth is

$$\sigma_k = \sum_{i \in k} \sum_{j=1}^{23} (|x_{i,1} - \mu_{k,1}|) \quad ,$$

where all  $C_j = 1$ , since no data is available regarding the relative economic worth of the respective functions. Thus, the excess functionality which results from developing a common interface to replace interfaces number 25 and 26 is

$$\begin{aligned}\sigma_{25} &= [(|1-1|) + [(|0-1|)] + \dots + (|0-0|)] \\ &\quad + [(|1-1|) + [(|1-1|)] + \dots + (|0-0|)] = 4 \quad .\end{aligned}$$

In order to make two unique interfaces common, at least one of the two will have to be changed. Typically, this change is in the form of providing some functions which are not necessary for one or both interfaces. These extra functions may manifest themselves by increasing interface weight, decreasing the clearance between connectors in the interface, and changing the physical size of the interface. On the other hand, the benefits of having to design and develop fewer interfaces may well outweigh these costs.

Because this is a Conceptual Design study, few specifics are known regarding design parameters and economic impact associated with the individual functions. The main objective of this commonality analysis is to establish whether there is enough commonality present among the interfaces to make the common berthing interface concept feasible and, if so, to make a preliminary definition. For these reasons, a simple objective function may be employed — one in which  $f_1(n,s)$  is linear in  $n$ . Let

$$Z(n,s) = K \cdot n + s$$

be the objective function to be minimized where  $K$  is the maximum number of excess functions which may be incurred in order to reduce the number of interfaces by one. While an economic worth assessment could be performed to determine  $K$ , a standard procedure in conceptual design studies is to vary controlling parameters over a range to examine the sensitivity of the solution. Therefore,  $K$  will be varied over the range from zero up to that level required for all interfaces to be replaced by a single common interface.

The commonality analysis methodology developed in Section III and IV was applied to the berthing interface data given in Table 11. A dendogram depicting the hierarchical clustering of the berthing interfaces is given in Figure 1. The vertical line shown at  $K=30$  defines the hierarchical clustering solution at that point. The partition defined by  $K=30$  consists of five common berthing interfaces and is given in Table 13. This partition was used as the initial partition for the switching algorithm, and the switching algorithm identified two changes which improved the solution. For  $K=30$ , the value of the objective function was reduced to -1131 from -1129 by making these switches. The final clustering solution for  $K=30$  is given in Table 14 with the two changes from Table 13 being underscored. Other partitions defined by other values of  $K$  may be used as the initial partition for the switching algorithm. For instance, at  $K=10$ , the switching algorithm makes no changes and at  $K=20$ , the switching algorithm makes two changes, reducing the objective function to -711 from -707.

A graph depicting the total excess functionality  $S$  versus the number of berthing interfaces  $N$  is given in Figure 2. The values for  $S$  were taken from the results of the hierarchical clustering portion of the clustering method. Since the switching algorithm made so few changes to the hierarchically formed partitions for  $K=10$ , 20, and 30, and had an insignificant impact on the value of the objective function, the hierarchically formed partitions should represent a very good approximation to a minimal value of  $S$ . Note from Figure 2 that the number of interfaces can be reduced from 48 to 13 while only incurring a total of 46 excess functions. Thus, at a price of one extra function per berth on average, the number of berths which must be developed is reduced by a factor of four. Likewise, at a price of two extra functions per berth on average, the number of berths which must be developed is eight — a reduction by a factor of six. If the number of berths is reduced to five, just over three extra functions per berth on average is required. It is these five common berths which are given in Table 14.

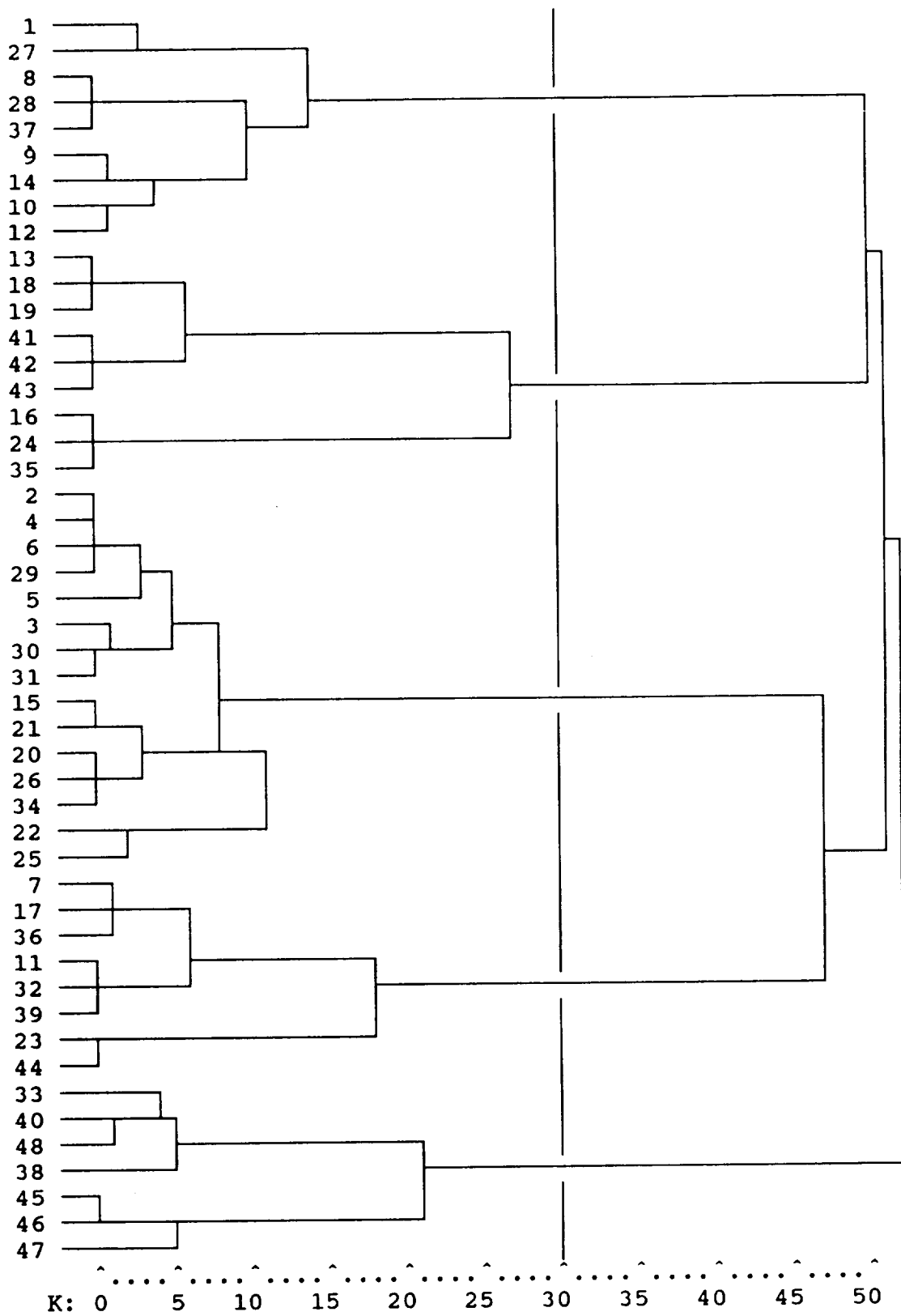


Figure 1. Hierarchical clustering of berthing interfaces.

TABLE 13. HIERARCHICALLY FORMED PARTITION (K=30)

Partition	Functions Required
1,8,9,10 12,14,27 28,37	<1,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0>
2,3,4,5 6,15,20 21,22,25 26,29,30 31,34	<1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0>
7,11,17 23,32,36 39,44	<1,1,1,1,1,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0>
13,16,18 19,24,35 41,42,43	<1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0>
33,38,40 45,46,47 48	<1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,1>

TABLE 14. FINAL CLUSTERING SOLUTION (K=30)

Partition	Functions Required
1,8,9,10 27,28,37	<1,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0>
2,3,4,5 6,15,20 21,22,25 26,29,30 31,34	<1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0>
7,11,17 23,32,36 39,44	<1,1,1,1,1,1,0,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0>
<u>12,13,14</u> 16,18,19 24,35,41 42,43	<1,1,1,0,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,0,0,0>
33,38,40 45,46,47 48	<1,0,1,0,1,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,1,1>



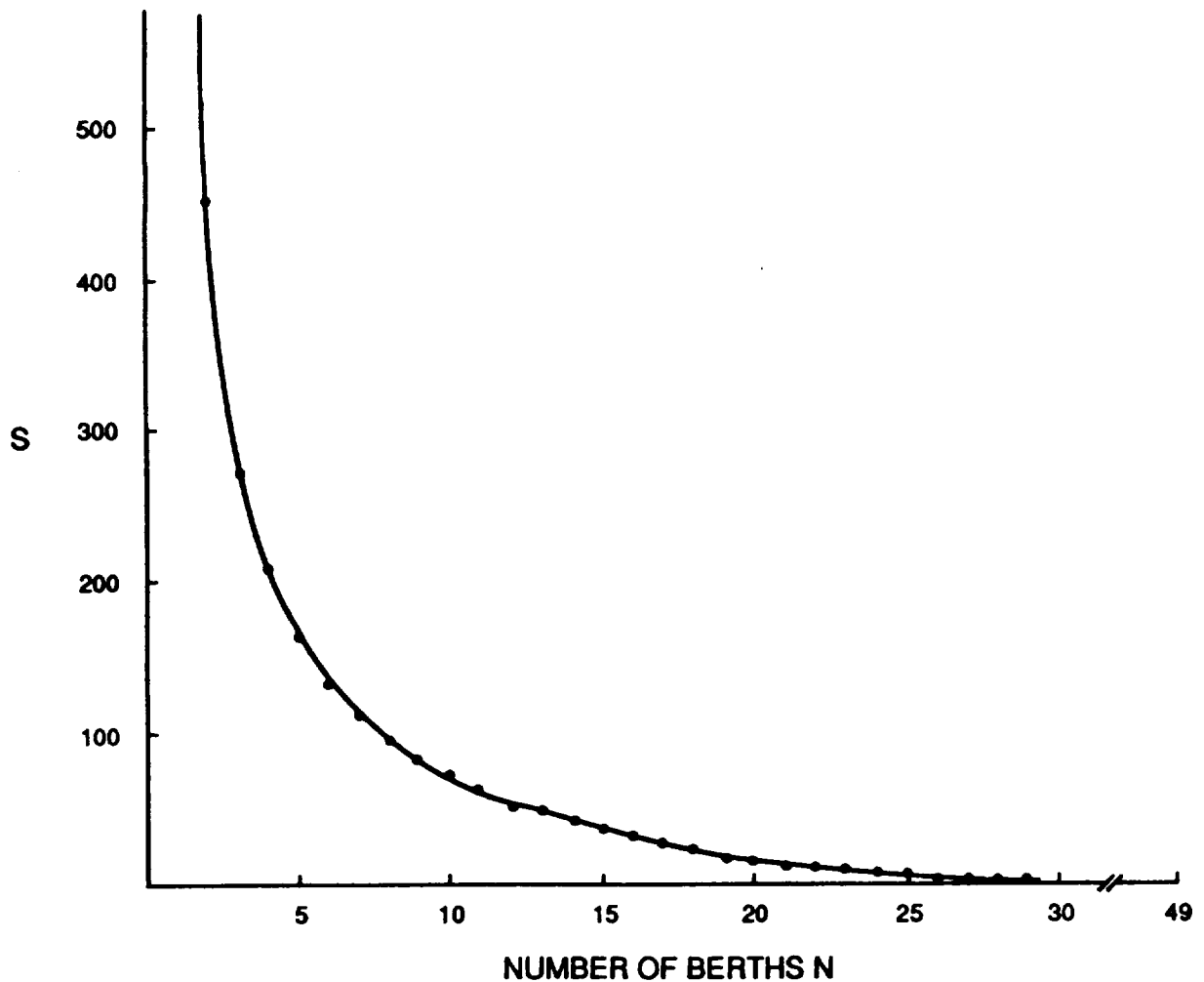


Figure 2. Variation of excess functionality with number of berths.

While further economic and technical data would be required to reach a definitive conclusion regarding the specification of common berthing interfaces, the potential for developing a small family of common interfaces appears excellent. Data needed to support further studies includes the relative costs of the 27 functions (they were assumed to be equal in this study) and the development cost of a module berthing interface. Since conclusions regarding the specification of common interfaces are not yet required, there is no justification to attempt to verify the optimality of the solution. However, optimality considerations would be required of follow-on studies, particularly commonality analyses during Detail Design which address design specification of the module berthing interface. The computer programs used for this commonality analysis are listed in Appendix B.

#### E. The Role of Commonality Analysis in Preliminary Design

During the Preliminary Design phase of a system, several candidate system concepts are studied further and one candidate system is selected as optimal. This selected concept is then further studied and optimized to permit definitive specification of the system. Prior to concept selection, the application of commonality is

analogous to that in Conceptual Design. However, since some additional definition of the concepts take place, the likelihood of identifying cost effective commonality applications increases.

After concept selection, commonality may be pursued to lower levels of definition. The subsystem configuration has been defined and the design effort begins to center on more detailed study of subsystem components. Data may be collected on the functional requirements of frequently used components such as electric motors, digital control units, power conditioners, structural supports, etc., and a plethora of interfaces. While the quality of the data may not support the definitive identification of an optimal solution, trends and guidelines may be identified for further study during Detail Design [29].

The common module definition study referenced in Section I is this type of commonality analysis. After the reference configuration of the space station was selected, this commonality study addressed the definition of a common module to serve as a building block for the five modules in the configuration [4]. This commonality analysis illustrated the feasibility and cost effectiveness of the common module approach as well as providing preliminary recommendations regarding the makeup of this common module.

Another such study addressed the development of common interfaces for the space station equipment rack [26]. After selection of a reference space station configuration, the nature of the module interior architecture was investigated. In a study very similar in concept to the common berthing interface study previously discussed, it was determined that the 44 unique rack utility interfaces in the space station laboratory module could be replaced by ten common utility interfaces. Since this commonality analysis was based on Preliminary Design data, the conclusion of ten common utility interfaces is necessarily preliminary as well. The conclusion drawn from the study was that common rack utility interfaces are cost effective, with the exact composition of the interfaces to be determined during Detail Design.

The Preliminary Design phase culminates in a definitive system specification. The functions and performance levels of the subsystem components have been established. While the nuts and bolts have not been specified, the basic approaches for the construction of all hardware and software are understood. Since the cost effectiveness of commonality is dependent upon functional similarity and the functions of all hardware and software have been established, those areas where commonality should be applied can be identified during Preliminary Design. Detail Design data will be needed to determine the exact nature of the commonality application and, in borderline cases, to determine the economic feasibility. Still, most commonality applications should be identified by the conclusion of Preliminary Design. The Systems Design Review should address specific recommendations for the development of common items based on the preliminary commonality analyses [21].

#### F. Commonality Analysis of Electric Motors (Part 1)

The scope and objectives of commonality analysis in Preliminary Design are much the same as in Conceptual Design. The principle difference is the kind of items addressed in each; in Conceptual Design, concentration is at the subsystem level while in Preliminary Design, commonality of lower level items such as electric motors may be addressed. Consider the listing of electric motors given in Table 15. These motors were identified during the Preliminary Design activity of the space station program. These motors are used to power fans, pumps, actuate valves and dampers,

TABLE 15. PRELIMINARY DESIGN DATA FOR  
ELECTRIC MOTORS

<u>Motor</u>	<u>Quantity</u>	<u>Power (watts)</u>
1	12	3.5
2	4	6.6
3	22	6.5
4	4	6.8
5	2	7.8
6	8	10.0
7	2	20.0
8	12	32.0
9	4	68.0
10	1	280.0
11	7	250.0
12	4	300.0
13	2	285.0
14	5	350.0
15	6	510.0
16	8	200.0
17	4	98.0
18	8	80.0
19	4	79.9
20	8	67.0
21	22	66.0
22	22	13.0
23	24	12.5

and other applications. As in the tankage example in Section IV, the feasibility relation is one of size; it is feasible to use a larger motor instead of a smaller motor. Where commonality between two motors is sought, the design characteristics of the motor with the larger power requirement become those of a common motor.

The commonality analysis methodology developed in Sections III and IV was applied to the electric motor data given in Table 15. A linear objective function of the form

$$Z(n,s) = K \cdot n + s$$

was employed. A dendrogram depicting the hierarchical clustering of the electric motors is given in Figure 3. The partition defined at  $K=300$  consists of nine types of motors and is given in Table 16. This partition was used as the initial partition for the switching algorithm, which made no changes. Thus, the hierarchically derived partition given in Table 16 is also the final clustering solution. The value of the objective function at  $K=300$  is 15,862.

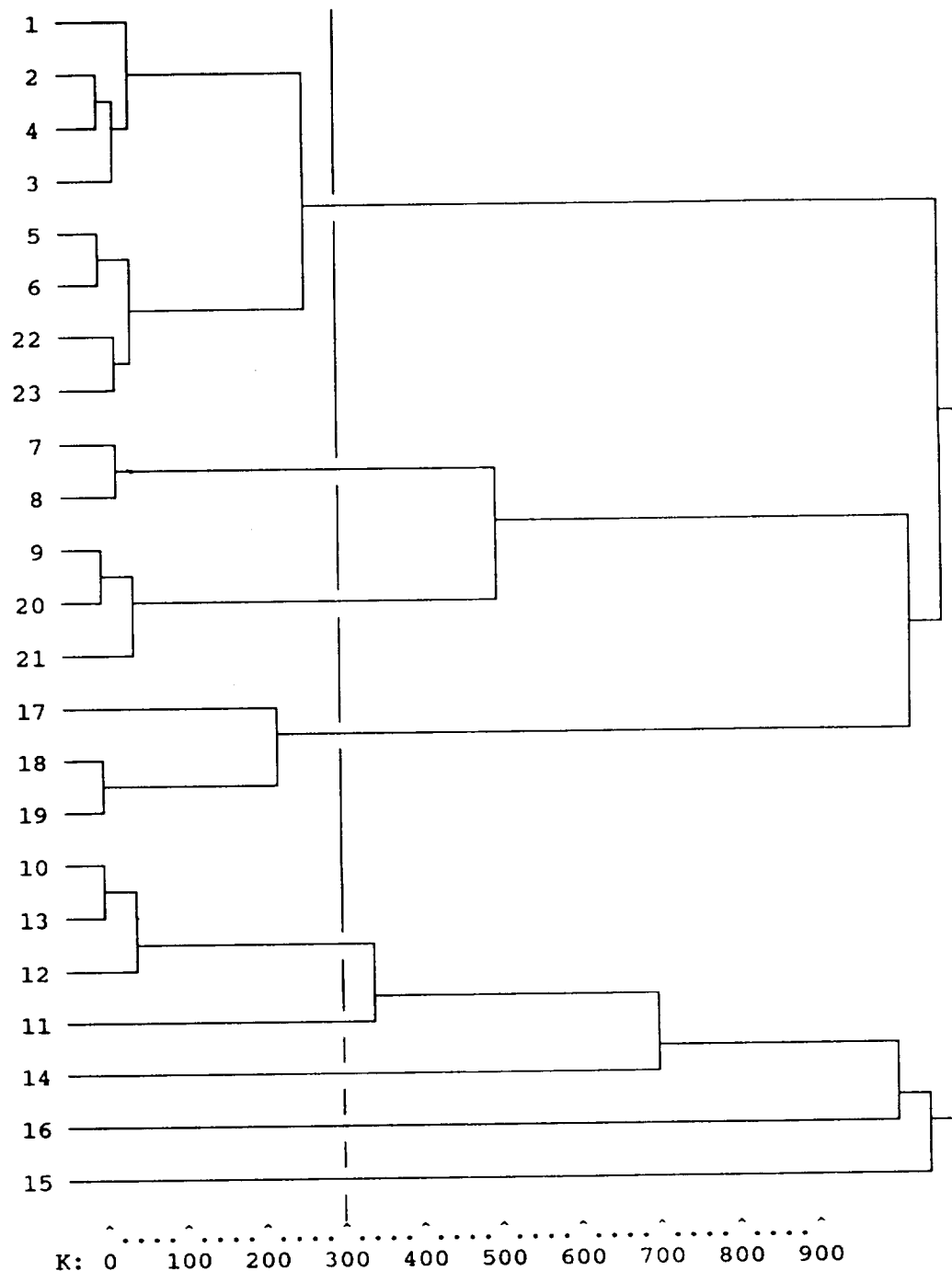


Figure 3. Hierarchical clustering of electric motors.

TABLE 16. HIERARCHICALLY FORMED PARTITION (K=300)

Partition	Total Quantity	Power (watts)
1,2,3,4,5,6,22,23	98	13.0
7,8	14	32.0
9,20,21	34	68.0
10,12,13	7	300.0
11	7	250.0
14	5	350.0
15	6	510.0
16	8	200.0
17,18,19	16	98.0

A graph depicting the total excess functionality  $S$  versus the number of different types of motors  $N$  is given in Figure 4. Recall that in this case,  $S$  defines the extra power requirements of the oversized motors used in common applications. From Figure 3, it is noted that at  $K=40$ , 13 different types of motors are required. From Figure 4, it may be noted that the excess power required to reduce the number of different types of motors from 23 to 13 is 134. Thus, for an increased power allocation of less than 1 percent, the number of different types of motors which must be developed is almost cut in half. Likewise, at  $K=300$ , the reduction to nine types of motors requires an extra 700 W of power, an increase of 5 percent. If the power allocation for electric motors can be increased by 10 percent, the number of motors which must be developed can be reduced to 7; at  $K=510$ , the 7 types of motors require an extra 1,550 W.

Further economic and technical data will be required to specify the exact nature of the commonality solution. However, since the number of different types of motors may be reduced significantly with marginal increases in power consumption, the feasibility of commonality is established. In regard to a preliminary definition of the commonality solution, the partitions given in Table 16 indicate that the smaller motors tend to be clustered, whereas the larger motors do not. Of course, this is attributable to the more substantial power requirement penalties which accompany making the larger motors common. Still, this study treated equally the cost savings due to making any motor common, and it is quite likely that the cost savings of making larger motors common would be proportionally greater than for smaller motors. This issue and others, including optimality considerations, will be addressed in the commonality analysis of electric motors performed during Detail Design. The computer programs used for this commonality analysis are listed in Appendix B.

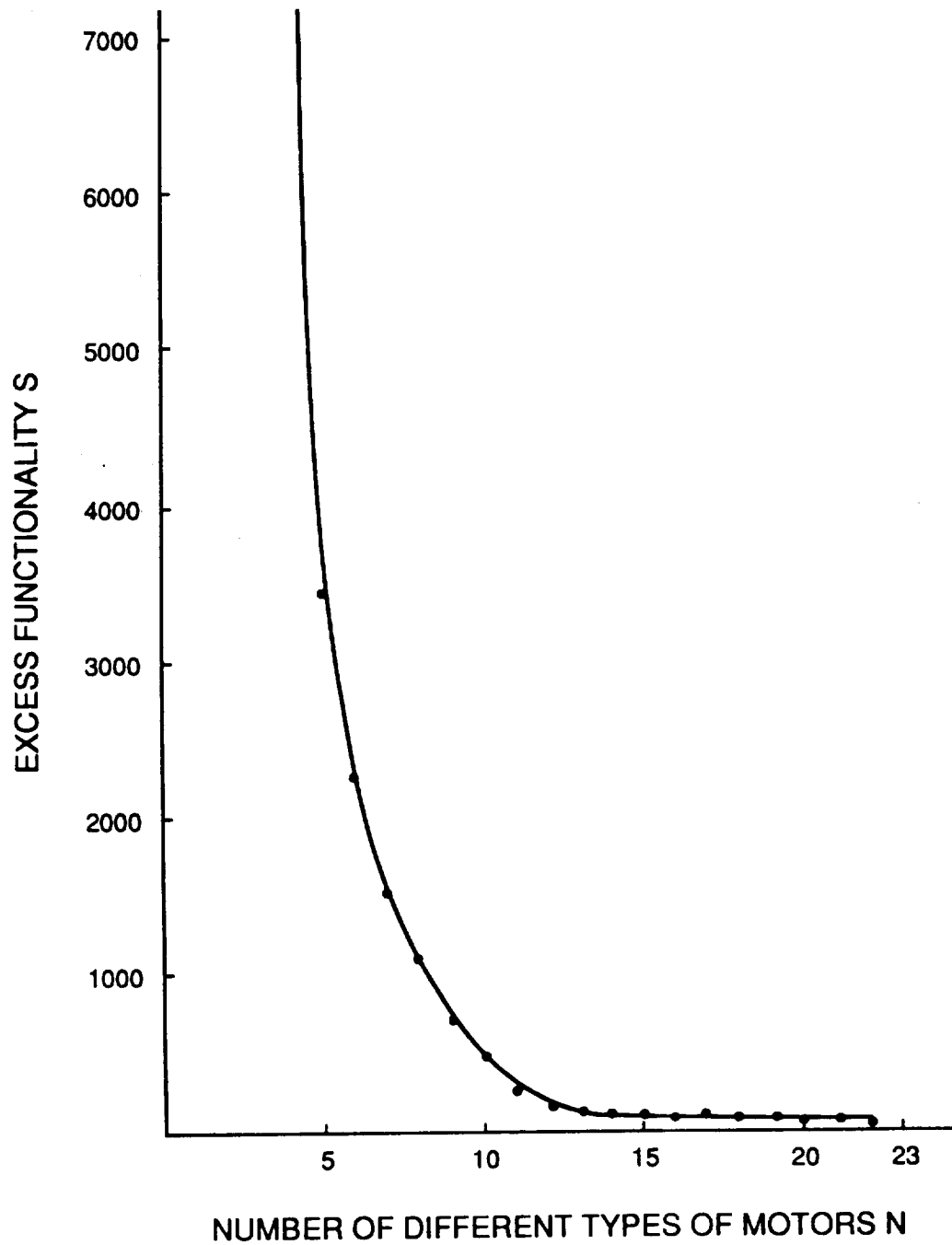


Figure 4. Variation of excess functionality with number of types of motors.

## G. The Role of Commonality Analysis in Detail Design

The functional specification and systems requirements developed during Preliminary Design evolve into manufacturing drawings and plans during the Detail Design phase. During this process, functional definition and allocation to the lowest levels marks the completion of the system analysis activity. The development of detailed specifications for manufacturing represents the integration of nuts and bolts into parts, parts into components, components into subsystems and so on to construct the system. The Detail Design activity for a common item does not differ from the Detail Design for any other item. The lone distinction of a common item from the design standpoint is that, in general, it must satisfy a broader range of functional requirements [31]. Thus, the role of commonality analysis in Detail Design consists of identifying the common items and their respective functional requirements. Once this task has been performed, commonality analysis is, in essence, complete.

The commonality analysis process in the early part of the Detail Design phase is analogous to that of Preliminary Design. As functional definition progresses to successively lower levels, additional commonality applications may be identified. By the completion of the system analysis activity, all information needed for the identification of commonality applications is complete and, hence, all commonality applications should have been identified. With the beginning of detailed specification development, data will become available permitting the exact nature of the previously identified commonality applications to be ascertained. Specific objective functions may be formulated using cost relationships such as those employed by the System Commonality Analysis Tool, discussed in Section II, with enough precision to permit the identification of optimal solutions.

Since the Equipment Design Review is scheduled near the beginning of the detailed specification development activity (shortly after the completion of system analysis), it can serve as a milestone for the commonality analysis activity. At this review, all commonality applications recommended for the program should be addressed. Subsequently developed detailed data will permit the exact specification of common items, and this activity should be completed with results addressed at the next milestone — the Critical Design Review. The system design is baselined at the Critical Design Review, and ideally the commonality analysis activity should be completed shortly after this review [31]. However, occasionally changes to the design baseline are necessary, and if these changes affect common items, a revised commonality analysis is required [32]. A design change for an application using a common item has the potential to affect the overall requirements for the common item itself and thus alter the cost effectiveness. This revised commonality analysis will define a revised optimal commonality solution in light of the design change.

In summary, the role of commonality analysis during Detail Design is initially one of extending the identification of commonality alternatives begun during Preliminary Design. The subsequent and final activity in commonality analysis utilizes the evolving Detail Design data to determine the optimal implementation of commonality in each of the previously identified areas. Commonality analysis for a system is completed during Detail Design and no new activities are required in the subsequent Development phase; revisions to commonality analyses will be required when downstream design changes affect common items.

## H. Commonality Analysis of Electric Motors (Part 2)

Recall the previous example of a Preliminary Design commonality analysis of electric motors for the space station. The space station program is currently in the early stages of Detail Design, and the data given in Table 17 has become available for the electric motor applications aboard the space station. Note that in comparison to Table 15, data is now available specifying the weight, volume, estimated R&D cost, and estimated unit production cost. Note also that there are now 27 different types of motors rather than 23 as before. In commonality analysis during Conceptual and Preliminary Design, the primary objective was to establish the feasibility of commonality with a secondary emphasis upon preliminary definition of a commonality solution. Now, with this additional economic and technical data, the objective of this commonality analysis is to provide specific recommendations regarding which motors should be developed in common.

The commonality analysis methodology developed in Sections III and IV was applied to the electric motor data given in Table 17. An objective function similar to that used in the ECLS Water Tankage example presented in Section IV was employed:

$$Z(n,s) = \sum_{k=1}^n [\mu_{k,4} - \mu_{k,5} \cdot \sum_{j=1}^{QN_k} j^{\ln(LC/100)/\ln(2)}] + s + C$$

where

$$C = \left( \sum_{i=1}^M QN_i \cdot x_{i,5} \right), \quad QN_k = \sum_{i \in k} QN_i,$$

and LC is the rate of learning.

The principle difference between this objective function and that used in the ECLS Water Tankage example is the incorporation of a production learning curve in this example. The hierarchically derived clustering solution is given in Table 18. This partition served as the initial partition for the switching portion of the clustering algorithm, which made no changes. Thus, the hierarchically derived partition given in Table 18 is also the final clustering solution. The value of the objective function for the partition defining the final clustering solution is 3417.34, a net reduction of 640.34 — approximately 15 percent — compared to the "no commonality" option. As in the Preliminary Design commonality analysis, the clustering occurs principally among the smaller motors.

The value of the objective function for various levels of commonality is depicted in Figure 5. This graph was constructed using results of the hierarchical clustering procedure, which was allowed to continue execution beyond the minimal value obtained for 14 clusters. The hierarchical partitions are being used as approximations to Z-minimal partitions. Note from Figure 5 that the objective function begins to increase rapidly for fewer than ten clusters, indicating that the excess functionality incurred grows very large. Thus, whereas in Phase B the possibility of as few as seven different types of motors was entertained, it has now been established that this level of commonality actually increases system cost.



TABLE 17. DETAIL DESIGN DATA FOR ELECTRIC MOTORS

Motor	Quantity	Power (watts)	Weight (lbs)	Volume (ft3)	R&D Cost (\$K)	Prod Cost (\$K/unit)
1	12	3.5	0.350	0.004	18.32	8.71
2	4	6.6	0.554	0.006	24.83	11.80
3	22	6.5	0.012	0.001	26.80	12.73
4	4	6.8	0.517	0.006	23.72	11.27
5	2	7.8	0.679	0.008	28.41	13.50
6	8	10.0	0.600	0.007	26.17	12.44
7	2	20.0	1.120	0.013	39.56	18.80
8	3	32.0	1.408	0.016	46.02	21.87
9	9	32.0	1.344	0.015	44.63	21.21
10	4	68.0	2.448	0.028	66.36	31.53
11	1	280.0	5.880	0.068	118.49	56.31
12	7	250.0	3.500	0.041	84.06	39.95
13	4	300.0	4.200	0.049	94.84	45.07
14	2	285.0	3.705	0.043	87.29	41.48
15	5	350.0	4.900	0.057	105.02	49.91
16	4	510.0	5.865	0.068	118.29	56.21
17	2	510.0	5.610	0.065	114.86	54.58
18	4	200.0	1.800	0.021	54.14	25.73
19	4	98.0	0.960	0.011	35.72	16.98
20	8	80.0	0.800	0.009	31.66	15.05
21	4	79.0	0.869	0.010	33.44	15.89
22	8	67.0	0.804	0.009	31.77	15.10
23	22	66.0	0.858	0.010	33.16	15.76
24	22	13.0	0.351	0.004	18.36	8.72
25	2	12.5	0.400	0.005	20.02	9.51
26	22	12.5	0.425	0.005	20.84	9.90
27	4	200.0	1.400	0.016	45.85	21.79

TABLE 18. HIERARCHICALLY FORMED PARTITION

Partition	Quantity	Attribute Vector
1	12	< 3.5, 0.35, 0.004, 18.32, 8.71 >
2,4,5,6	18	< 10.0, 0.68, 0.008, 28.41, 13.50 >
3	22	< 6.5, 0.01, 0.001, 26.80, 12.73 >
7,8,9	14	< 32.0, 1.41, 0.016, 46.02, 21.87 >
10	4	< 68.0, 2.45, 0.028, 66.36, 31.53 >
11,15	6	< 350.0, 5.88, 0.068, 118.49, 56.31 >
12	7	< 250.0, 3.50, 0.041, 84.06, 39.95 >
13,14	6	< 300.0, 4.20, 0.049, 94.84, 45.07 >
16,17	6	< 510.0, 5.86, 0.068, 118.29, 56.21 >
18,27	8	< 200.0, 1.80, 0.021, 54.14, 25.73 >
19	4	< 98.0, 0.96, 0.011, 35.72, 16.98 >
20,21	12	< 80.0, 0.87, 0.010, 33.44, 15.89 >
22,23	30	< 67.0, 0.86, 0.010, 33.16, 15.76 >
24,25,26	46	< 13.0, 0.42, 0.005, 20.84, 9.90 >

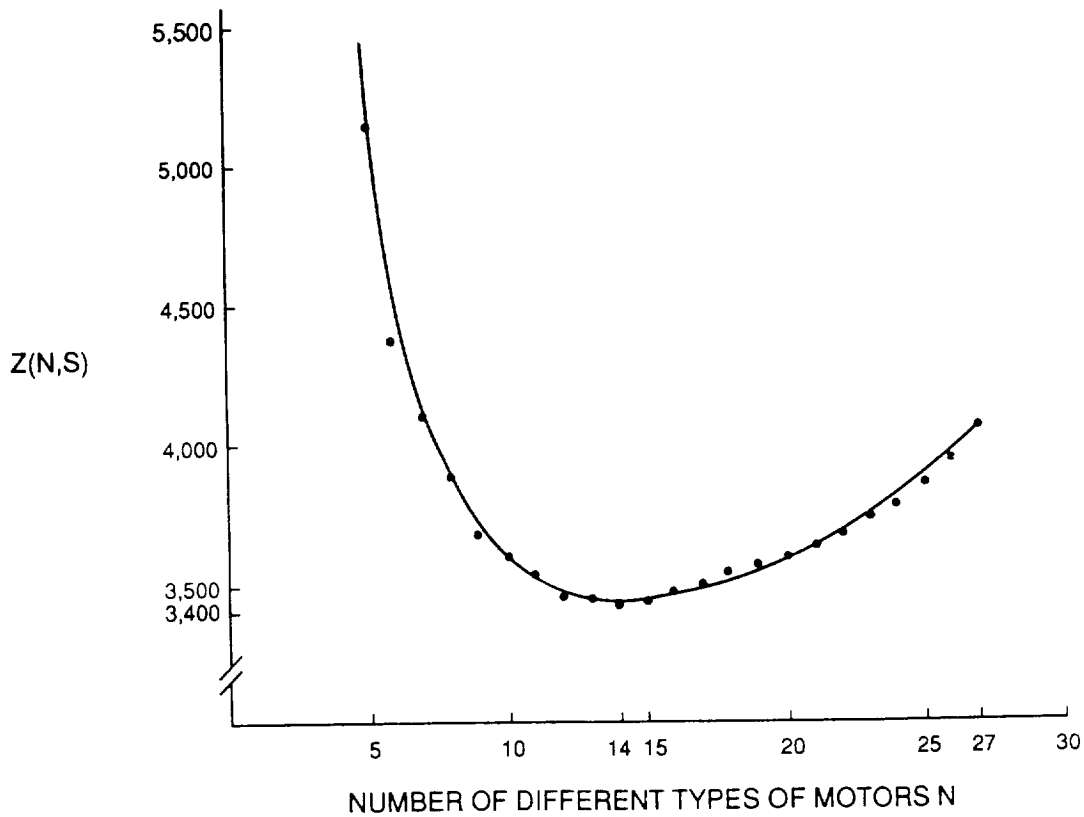


Figure 5. Cost benefit of various levels of commonality.

The clustering solution given in Table 18 cannot be verified as a Z-minimal partition. Unfortunately, such a large number of the sufficiency inequalities are violated for the commonality solution that it is not possible to enumerate all possible partitions, and thus the commonality solution may not be verified as Z-minimal. Since it is not possible to identify a Z-minimal partition for  $N=14$ , the optimal solution cannot be identified. A description of the application of the Z-minimal sufficient conditions to the partition of Table 18 is given in Appendix C.

The final clustering solution is comprised of 14 different types of motors. However, it must be noted that if research and development costs were revised upward in a subsequent commonality analysis cost update, a commonality solution of fewer different types of motors would be the result. Likewise, if the weight, volume, and power parameters are revised upward, more different types of motors will be specified in the commonality solution. While the partition given in Table 18 does represent a definitive commonality solution, this commonality analysis would require revision as further data — such as duty cycle and mean time between failure — becomes available and as a result of changes in existing motor specifications and cost estimates. However, any revision would be of the same genre as this commonality analysis. While this commonality analysis differed markedly from the Phase B commonality analysis presented earlier, revisions to this commonality analysis would differ mainly in number of attributes. The computer programs used for this commonality analysis are listed in Appendix B.

## VI. CONCLUSIONS AND RECOMMENDATIONS

### A. Conclusions

The purpose of the research documented in the previous chapters was the creation of a commonality analysis methodology. By formulating commonality analysis as a partitioning problem, clustering techniques may be applied. The use of clustering techniques allows solution to commonality analysis problems by a means other than complete enumeration, which is not feasible in most commonality analyses. The optimality of the clustering solution may be verified using the sufficiency conditions developed in Section IV.

In summary, three conclusions are drawn from previous discussion:

- 1) The methodology is adaptable to the requirements of commonality analysis within each of the design phases in the system development cycle.
- 2) The hierarchical clustering portion of the methodology appears to give very good results and in general provides an excellent approximation to Z-minimal partitions. In the commonality analysis problems used for examples, the switching portion of the algorithm made zero or few changes to the hierarchically derived partition and had a negligible impact on the value of the objective function. Where the Z-minimal sufficient conditions could be successfully applied, the hierarchical partitions were verified as Z-minimal partitions.
- 3) The sufficient conditions developed to verify Z-minimal partitions are not in general met by Z-minimal partitions. However, in some cases they may be used to pare the solution space to a number of alternatives for which enumeration may be practically employed. While this technique is laborious, it can represent a substitute for complete enumeration, which is generally not feasible.

The methodology developed and illustrated in the preceding chapters represents an initial step toward the definition of a generalized commonality analysis methodology. The clustering techniques employed in this methodology are well established and general in scope. The straightforward application of this methodology to representative commonality analyses from the various system design phases indicates a broad range of application. Rather than simply providing a definitive solution for a particular type of problem, the successful development and application of this methodology shows the utility of the clustering approach for commonality analysis problems in general.

## B. Recommendations

As discussed in the previous section, the hierarchical clustering technique appears to produce good approximations to Z-minimal partitions in general. Since the data on which even phase C/D commonality analyses are based is preliminary and thus subject to change, the nature of the optimal solution is likewise subject to change. For aerospace systems in particular, the specifications and cost estimates are not concrete until the item is delivered to the customer. Thus, the effort involved in implementing the switching techniques and verifying the solution using Z-minimal sufficient conditions may not be warranted. Further study on how well the hierarchically derived partitions approximate Z-minimal partitions for typical commonality analysis problems is needed to support any conclusion in this area.

In this research, concentration has been focused on arriving at the optimal solution, where the optimal solution is that partition producing the minimum value of the objective function. Due to the uncertainty inherent in commonality analysis data, the optimal solution may need to be defined in another manner. Rather than producing the minimum value of the objective function, an optimal solution could be defined as that partition producing a near minimum value of the objective function but somewhat insensitive to fluctuations in the commonality analysis data. If an optimal solution is defined in this manner, the likelihood of data revisions affecting the commonality solution would be decreased. A study addressing the weighting of proximity to the minimum value of the objective function versus degree of robustness of the partition is needed to support any conclusion in this area.

Finally, recall that the principle objective during Phase A and Phase B commonality analyses is to establish the feasibility of commonality. In general, these studies are parametric in nature in that the feasibility of commonality depends upon one or more parameters such as excess power or additional functions. While figures such as dendograms and plots are useful devices for depicting how commonality varies with parameters, the feasibility of commonality must still be deduced. One potential method for defining the feasibility of commonality would be to develop a statistical measure to estimate the likelihood of commonality for various values of a parameter. This statistic would be analogous to existing measures in cluster analysis which are utilized to define the number of clusters which exist in a data set [33]. A study in which the application of these methods to Phase A and Phase B commonality analysis data sets is needed to support any conclusion in this area.

The three aforementioned studies are recommended as immediate extensions of the research documented in this report. There are certainly others, such as the development of more powerful and practical optimality conditions. However, the outcomes of the preceding studies would have a definite impact on the scope of a

study addressing optimality conditions. This dissertation represents one step, and the above studies three more. After these, many steps will remain to be taken before a methodology for commonality analysis is fully defined and integrated into the system development process.



**APPENDICES**

**PRECEDING PAGE BLANK NOT FILMED**





# APPENDIX A. SELECTED PROOFS AND ILLUSTRATIONS

## A. Monotonicity Property of $\sigma$

From Section IV, the excess functionality for a single cluster is given as

$$\sigma_k = \sum_{i=1}^n QN_i \left\{ \sum_{j=1}^m C_j |x_{i,j} - \mu_{k,j}| \right\}, \quad x_i \in \text{cluster } k, \quad i = 1, 2, \dots, n.$$

Given two items with attribute vectors  $\underline{x}_a$  and  $\underline{x}_b$ , the excess functionality  $\sigma_{ab}$  is constant for all possible values of  $\underline{\mu}_{ab}$ . The excess functionality  $\sigma_{ab}$  is invariant to feasible values of the mean attribute vector  $\underline{\mu}_{ab}$  since

$$x \leq \mu_{ab}, \quad j \leq y$$

where

$$x = \text{Minimum}\{\underline{x}_{a,j}, \underline{x}_{b,j}\}$$

$$y = \text{Maximum}\{\underline{x}_{a,j}, \underline{x}_{b,j}\},$$

and because the excess functionality  $\sigma$  defined above is a summation of absolute differences between items and the mean attribute vector  $\underline{\mu}_{ab}$ . This invariance of  $\sigma$  to  $\underline{\mu}$  holds for any number of items in the cluster, where the relations  $x$  and  $y$  are extended to include the attribute vectors of all items.

From the formulation of  $\sigma$  above, the quantity  $\sigma_k$  is always nonnegative. For two items  $a$  and  $b$ ,  $\sigma_a$  and  $\sigma_b$  are zero. Consider the clustering of items  $a$  and  $b$ . Let

$$\sigma_{ab} = \sigma_a + \sigma_b + \delta(a,b)$$

where  $\delta(a,b)$  is a real number. In this case, since  $\sigma_a$  and  $\sigma_b$  are zero, then

$$\sigma_{ab} = \delta(a,b)$$

and thus  $\delta(a,b)$  is nonnegative since  $\sigma_{ab} \geq 0$ . Consider the addition of a third item  $c$  to cluster  $ab$ , yielding the quantity  $\sigma_{abc}$  of excess functionality. Then

$$\begin{aligned}
\sigma_{abc} &= \sigma_{ab} + \sigma_c + \delta(c,ab) \\
&= \sigma_{ab} + \delta(c,ab) \\
&= \delta(a,b) + \delta(c,ab)
\end{aligned}$$

where  $\delta(c,ab)$  is a real number. Now,

$$\underline{x}_a - \underline{\mu}_{abc} \geq \underline{x}_a - \underline{\mu}_{ab}$$

since the feasible values for  $\underline{\mu}_{abc}$  can include values not feasible for  $\underline{\mu}_{ab}$ ; likewise for  $\underline{x}_b$ . Hence, if  $\sigma'_{ab}$  denotes the excess functionality where  $\underline{\mu}_{abc}$  is used as the mean attribute vector, then

$$\sigma'_{ab} \geq \sigma_{ab}.$$

Since the absolute deviation of  $\underline{x}_c$  to  $\underline{\mu}_{abc}$  is at least zero, then

$$\sigma_{abc} \geq \sigma_{ab}$$

and thus  $\delta(c,ab)$  is also nonnegative. Likewise,

$$\sigma_{abcd} \geq \sigma_{abc}$$

and

$$\sigma_{abcd} = \delta(a,b) + \delta(c,ab) + \delta(d,abc)$$

and so on as additional items are included in the cluster. Thus, in general

$$\sigma_k = \delta(u,v) + \delta(w,uv) + \delta(y,uvw) + \dots$$

for all items which are members of cluster  $k$  including items  $u$ ,  $v$ ,  $w$ , and  $y$ , where all  $\delta$  are nonnegative. Therefore, the excess functionality will either remain the same or increase with the incorporation of additional items into a cluster.

## B. Order Independence of $\sigma$

From prior discussion, for four items a, b, c, and d, the excess functionality

$$\begin{aligned}\sigma_{abcd} &= \sigma_{abc} + \delta(d, abc) \\ &= \sigma_{ab} + \delta(c, ab) + \delta(d, abc) \\ &= \delta(a, b) + \delta(c, ab) + \delta(d, abc) \quad .\end{aligned}$$

Also from prior discussion,  $\sigma$  is invariant in  $\underline{\mu}$ . From equation (2), the value of  $\sigma$  is dependent solely on the membership of a cluster. Then,

$$\begin{aligned}\sigma_{abcd} &= \sigma_{bcd} + \delta(a, bcd) \\ &= \delta(b, c) + \delta(d, bc) + \delta(a, bcd) \\ &= \sigma_{acd} + \delta(b, acd) \\ &= \sigma_{abd} + \delta(c, abd)\end{aligned}$$

and so on.

Since  $\delta(u, v)$  defines the increase in  $\sigma$  which results from the addition of item v to cluster u, then

$$\delta(u, v) = \sigma_{uv} - \sigma_u \quad .$$

For two clusters k and l each comprised of one or more items, since

$$\sum_{i \in k} | \underline{x}_i - \underline{\mu}_{kl} | \geq \sum_{i \in k} | \underline{x}_i - \underline{\mu}_k |$$

and

$$\sum_{i \in l} | \underline{x}_i - \underline{\mu}_{kl} | \geq \sum_{i \in l} | \underline{x}_i - \underline{\mu}_l |$$

then

$$\sigma_{kl} \geq \sigma_k + \sigma_l \quad .$$

Thus,  $\delta(k,l)$  may be used to define the increase in  $\sigma$  which results from the merger of any two clusters:

$$\sigma_{kl} = \sigma_k + \sigma_l + \delta(k,l)$$

where  $\delta(k,l) \geq 0$ . Then for four items a, b, c, and d,

$$\begin{aligned}\sigma_{abcd} &= \sigma_{ab} + \sigma_{cd} + \delta(ab,cd) \\ &= \delta(a,b) + \delta(c,d) + \delta(ab,cd) \\ &= \delta(a,c) + \delta(b,d) + \delta(ac,bd) \\ &= \delta(a,d) + \delta(b,c) + \delta(ad,bc) \\ &\quad \cdot \\ &\quad \cdot \\ &= \delta(a,b) + \delta(c,ab) + \delta(d,abc) \quad .\end{aligned}$$

### C. Lower Bound on Increase in Excess Functionality

If the three objects a, b, and c are then combined to form a single group  $\{[a,b,c]\}$ , the value of the objective function is  $Z_{init} + D(a,b) + D(c,ab)$ . As shown earlier, the order in which members entered the cluster is of no consequence. Thus, in the case of a cluster of three objects a, b, and c,

$$D(a,b) + D(c,ab) = D(a,c) + D(b,ac) = D(b,c) + D(a,bc) \quad .$$

Likewise, from previous discussion,

$$\delta(a,b) + \delta(c,ab) = \delta(a,c) + \delta(b,ac) = \delta(b,c) + \delta(a,bc)$$

and since  $\delta$  is nonnegative,

$$\delta(a,b) + \delta(c,ab) \geq \text{Maximum}\{\delta(a,b), \delta(a,c), \delta(b,c)\} \quad . \quad (9)$$

From equations (4) and (9), a general expression for the minimum excess functionality which results from the commonality of the items in a particular cluster may be given as

$$\sigma_k \geq \text{Maximum}\{\delta(u,v)\} \quad (10)$$

for all items  $u,v \in$  cluster  $k$ . Using equations (5) and (9),

$$\sigma_k + \sigma_l + \delta(k,l) \geq \text{Maximum}\{\delta(u,v)\} \quad (11)$$

for all items  $u,v$  which are members of either cluster  $k$  or cluster  $l$ , may be used to infer a lower bound on the increase in  $S$  and, hence, the change in the objective function  $Z$  which results.

#### D. Proof of an S-Minimal Partition

Assume that the partition  $\{(abc),(def),(ghij)\}$  of ten objects into three groups satisfies the following inequalities:

1.  $\sigma_{abc} + \sigma_{def} \leq \text{Minimum}\{\delta(a,d), \delta(a,e), \dots \delta(c,f)\}$
2.  $\sigma_{abc} + \sigma_{ghij} \leq \text{Minimum}\{\delta(a,g), \delta(a,h), \dots \delta(c,j)\}$
3.  $\sigma_{def} + \sigma_{ghij} \leq \text{Minimum}\{\delta(d,g), \delta(d,h), \dots \delta(f,j)\}$

in accordance with the sufficiency condition of equation (15) of Section IV. Without loss of generality, let  $\delta(a,e)$ ,  $\delta(a,i)$ , and  $\delta(e,i)$  be the minimums in the preceding inequalities.

Consider the switch of item  $e$  to cluster  $abc$ . Then

$$\begin{aligned} \sigma_{abce} &\geq \text{Maximum}\{\delta(a,b), \delta(a,c), \delta(a,e), \dots \delta(c,e)\} \\ &\geq \delta(a,e) \end{aligned}$$

and, since  $\delta(a,e) \geq \sigma_{abc} + \sigma_{def}$  from inequality 1,

$$\sigma_{abce} \geq \sigma_{abc} + \sigma_{def}.$$

Denote the excess functionality of this partition  $S'$ . Then

$$\begin{aligned} S' &= \sigma_{abce} + \sigma_{df} + \sigma_{ghij} \\ &\geq \sigma_{abc} + \sigma_{def} + \sigma_{df} + \sigma_{ghij} \end{aligned}$$

and, since  $\sigma_{ghij}$  is unchanged and  $\sigma_{df}$  is nonnegative,

$$S' \geq S .$$

Since  $\delta(a,e)$  was the minimum, any other partition combining items of clusters abc and def cannot reduce S. Likewise, partitions combining items of clusters abc and ghij or of clusters def and ghij cannot reduce S. Next consider the switch of items e and i to cluster abc. Now,

$$\sigma_{abcei} = \sigma_{abc} + \delta(e,i) + \delta(ei,abc) ,$$

and since  $\delta(e,i) \geq \sigma_{def} + \sigma_{ghij}$ , then

$$\sigma_{abcei} \geq \sigma_{abc} + \sigma_{def} + \sigma_{ghij} + \delta(ei,abc) .$$

The excess functionality of this partition is

$$\begin{aligned} S' &= \sigma_{abcei} + \sigma_{df} + \sigma_{ghj} \\ &\geq \sigma_{abc} + \sigma_{def} + \sigma_{ghij} + \delta(ei,abc) + \sigma_{df} + \sigma_{ghj} . \end{aligned}$$

Since  $\delta(ei,abc)$ ,  $\sigma_{df}$ , and  $\sigma_{ghj}$  are nonnegative, again

$$S' \geq S .$$

Likewise, any partition combining objects from all three clusters cannot reduce S.

#### E. Illustration of Bounds on $\alpha$

Let the research and development cost for item u be denoted  $R\&D(u)$  and let the manufacturing cost savings which result from the quantity of item u be denoted  $MAN(u, QN_u)$ . Let manufacturing cost savings which result from making item u common with item v be denoted  $MAN(u, QN_u + QN_v)$ . For a cluster k formed from two items u and v, the lower bound on the net cost savings,  $\alpha_k$ , for the cluster is

$$\begin{aligned} |\alpha_k| &\geq \text{Minimum}\{R\&D(i)\} + MAN(i, QN_u + QN_v) \\ &\geq \text{Minimum}\{R\&D(i)\} + \text{Minimum}\{MAN(i, QN_i)\} . \end{aligned}$$

Likewise,

$$|\alpha_i| \leq \text{Maximum}\{R\&D(i)\} + \text{Maximum}\{MAN(i, QN_k)\}$$

for all  $i \in k$ , is the maximum contribution of any one item toward  $\alpha_k$ . Thus, the above equations define an upper bound on  $\alpha_i$  and a lower bound on  $\alpha_k$ , where the items  $i$  comprise cluster  $k$ .

To illustrate the foregoing inequalities, consider the cost savings which result from the merger of tanks 6 and 7 given in Table 2. Recall from two examples in Section IV that  $D(6,7) = -36$  while  $\sigma_{6,7} = 10$ . No learning curve was assumed in the production of tanks, so  $MAN(i, QN_i)$  is zero for all tanks and the sole source of cost savings is elimination of research and development activities. From Table 2,

$$x_{6,3} = 49 \quad \text{and} \quad x_{7,3} = 46$$

are the R&D costs for tanks 6 and 7, respectively. The cost to develop these tanks separately is then  $x_{6,3} + x_{7,3} = 95$ . By equation (18), the cost savings which result from the common tank are at least \$46,000 since

$$|\alpha_k| \geq \text{Minimum}\{49, 46\} = 46 \quad .$$

Likewise, the individual contributions of tank 6 and tank 7 toward this cost savings are at most \$49,000 since

$$|\alpha_i| \leq \text{Maximum}\{49, 46\} = 49 \quad .$$

Since the R&D cost from a previous example for the common tank,  $\mu_{6,3}$ , is 49, the cost savings which result from making tanks 6 and 7 common is

$$\alpha_{6,7} = -46 \quad .$$

Note that both of the above inequalities hold for  $\alpha_{6,7}$ .

#### F. Cost Impact of Switching Items Between Clusters

Given two clusters  $k$  and  $l$  each composed of two or more items, let an item  $u$  be switched from cluster  $k$  to cluster  $l$ . The maximum impact on  $f_1(N, S)$  is then the

maximum contribution of item  $u$  to  $\alpha_1$  minus the minimum remaining cost savings in cluster  $k$ ,  $\alpha_k$ :

$$\alpha^*(k) = [\text{Maximum}\{R\&D(u)\} + \text{Maximum}\{\text{MAN}(v, QN_v + QN_1)\}] \\ - [\text{Minimum}\{R\&D(w)\} + \text{Minimum}\{\text{MAN}(y, QN_y)\}]$$

for all  $u, v, w, y \in$  cluster  $k$ . In general, for two clusters  $k$  and  $l$  each composed of two or more items,

$$\alpha^*(k) = [\text{Maximum}\{R\&D(u)\} + \text{Maximum}\{\text{MAN}(v, QN_k + QN_l)\}] \\ - [\text{Minimum}\{R\&D(w)\} + \text{Minimum}\{\text{MAN}(y, QN_y)\}]$$

for all  $u, v, w, y \in$  cluster  $k$ , is the maximum reduction in  $f_1(N, S)$  which may be incurred from switching one or more, up to  $n-1$ , items from cluster  $k$  to cluster  $l$ . If cluster  $k$  is composed of only one item, then  $\alpha^*(k)$  is zero since it is not possible to switch an item from cluster  $k$  to another cluster without reducing  $N$ .



## APPENDIX B. COMPUTER PROGRAMS FOR COMMONALITY ANALYSIS

### A. General

The following computer programs were used for the numerical commonality analysis examples given in Sections IV and V. Both programs are written in Turbo Pascal for execution on an IBM Personal Computer. The first program implements the hierarchical clustering portion of the commonality analysis methodology, and the second program implements the switching portion. In this case, the programs are set up for the final example, "Commonality Analysis of Electrical Motors (Part 2)." The programs may be adapted to the other numerical examples by changing the constants and the function Distance. The program outputs are given following the program listings.

### B. Hierarchical Clustering Program

```
program commonality;

uses printer, crt;

{ This program pares the commonality solution space by
  employing hierarchical clustering methods.}

const
    K := 0.0;           {constants}
    NumItems = 27;      {weighting if linear objective function}
    NumAtts = 5;        {number of different sizes of motors}
    LC = 0.85;          {number of attributes for each motor}
                       {85% learning curve}

type
    Data = string;
    DataFile = text;
    Attarray = array[1..NumAtts] of real;
    ShString = string[35];
    DataRecord = record
        Mnum           : Data;
        Mquan          : integer;
        Mspecs         : Attarray;
    end;
    DataArray = array[1..NumItems] of DataRecord;
    SimilarityMatrix = array[1..NumItems,0..8] of real;
    DisArray = array[1..2,1..8] of real;           (8=NumAtts+3)
    Partition = record
        PartArray:array[1..NumItems] of integer;
    end;

var
    SimMatrix      : SimilarityMatrix;
    MotorNum       : ShString;
    MotorQuan      : integer;
    MotorSpecs     : Attarray;
    I,J,II,NumClus : integer;
    Flag           : integer;
```

```

Dist, Cost, LRN      : real;
Dfile, ComFile       : DataFile;
DataLine             : Data;
FileSpec             : string[80];
MotorData            : DataArray;
MinDist              : real;
Mini, Mink           : integer;
charv                : char;
Q, S, f              : integer;
DisMat               : DisArray;

```

```

procedure decode(Line : Data; var Mnum : ShString;
                 Var Mquan : integer; Var Mspecs :
Attarray);
(unpacks the ASCII text lines and loads variables per line )

```

```

var
  Darray      : array[1..7] of
ShString; (NumAtts+2)
  I, J, code, Marker : integer;
  V             : real;
  dummy        : ShString;

begin J := 1; Marker := 0;
for I := 1 to length(Line) do
  {identify substrings by comma separators}
  if ((Line[I] = chr(44)) or (I = length(Line))) then
    begin { unpack substrings }
      if Line[I] = chr(44) then
        Darray[J] := copy(Line, Marker+1, I-Marker-1)
        else Darray[J] := copy(Line, Marker+1, I-Marker);
        Marker := I;
        J := J + 1;
      end;
    Mnum := Darray[1]; {load variables}
    val(Darray[2], Mquan, code);
    for J := 3 to NumAtts+2 do
      begin
        dummy := Darray[J];
        if dummy[length(dummy)] = chr(32) then
          begin {write a '0' in Darray[J] if it has only blanks}
            dummy[length(dummy)] := chr(48);
            Darray[J] := dummy;
          end;
        val(Darray[J], V, code);
        if code <> 0 then
          begin
            writeln('Decoding error in string location ', code);
            writeln('of string ', J, ' consisting of ', Darray[J]);
            halt;
          end
        else Mspecs[J-2] := V;
        end;
      end;
    end;
end;

```

```

procedure getfile(FileSpec : string;
                  var MotorData : DataArray);

var
    MotorNum      : ShString;
    MotorQuan     : integer;
    MotorSpecs    : Attarray;
    I,J,Recnum    : integer;
    Dfile         : text;
    Drecord       : string;

begin
    assign(Dfile, FileSpec);
    reset(Dfile);
    Recnum := 0; while not eof(Dfile) do
        begin
            Recnum := Recnum + 1;
            readln(Dfile,Drecord);
            decode(Drecord, MotorNum, MotorQuan, MotorSpecs);
            with MotorData[Recnum] do
                begin
                    Mnum := MotorNum;
                    Mquan := MotorQuan;
                    for J := 1 to NumAtts do Mspecc[J] := MotorSpecs[J];
                end;
            end;
        end;
    close(Dfile);
end;

function max(x,y:real):real;
begin
    if x < y then max := y else max := x;
end;

function sigma(Dmat : DisArray) : real;
{ tally excess functionality to cluster I and J }

var
    QNi,QNj,sg    : real;
    I              : integer;
    Maxi,Cj       : array[1..NumAtts] of real;

begin
    sg := 0.0;
    QNi := Dmat[1,Q];
    QNj := Dmat[2,Q];
    Cj[1] := 0.5; (500 dollars per watt)
    Cj[2] := 2.8; (2800 dollars per pound)
    Cj[3] := 10.0; (10,000 dollars per cubic foot)
    Cj[4] := 0.0;
    Cj[5] := 1.0;

```

```

for I := 1 to NumAtts do Maxi[I]:=max(Dmat[1,I],Dmat[2,I]);
  (define mean attribute vector above) for I := 1 to
NumAtts do
  sg := sg + QNi*Cj[I]*(abs(Dmat[1,I] - Maxi[I]))
    + QNj*Cj[I]*(abs(Dmat[2,I] - Maxi[I]));
  (sigma is increase in excess functionality)
sigma := sg;
end;

function alpha(Dmat : DisArray):real;

var
  QNk, RD, Prod, LCS      : real;
  I, QN, QNii, QNji       : integer;
  QNi, QNj, LCSi, LCSj    : real;

begin
  QNi := Dmat[1,Q];  QNii := round(QNi);
  QNj := Dmat[2,Q];  QNji := round(QNj);
  QNk := Dmat[1,Q] + Dmat[2,Q];
  QN  := round(QNk);
  RD := Dmat[1,4] + Dmat[2,4] - max(Dmat[1,4],Dmat[2,4]);
  LCSi := 0.0;
  for I := 1 to QNii do LCSi := LCSi + exp(LRN*ln(I));
  LCSi := (QNi-LCSi)*Dmat[1,5];
    (learning curve savings included in alpha i)
  LCSj := 0.0;
  for I := 1 to QNji do LCSj := LCSj + exp(LRN*ln(I));
  LCSj := (QNj-LCSj)*Dmat[2,5];
    (learning curve savings included in alpha j)
  Prod := max(Dmat[1,5],Dmat[2,5]);
  LCS := 0.0;
  for I := 1 to QN do LCS := LCS + exp(LRN*ln(I));
  alpha := -(RD + Prod*(QNk - LCS) - LCSi - LCSj);
    (f1(N,S) is cost savings)
end;

function objective_function(Alf,Sig:real):real;
begin
  objective_function := Alf + Sig;
end;

function Distance(Dmat : DisArray) : real;
  (assesses the distance between clusters I and J)

var
  Unique, Common, Alf, Sig      : real;

begin Unique := objective_function(Dmat[1,f],Dmat[1,S])
  + objective_function(Dmat[2,f],Dmat[2,S]);
  Alf := alpha(Dmat) + Dmat[1,f] + Dmat[2,f];
  Sig := sigma(Dmat) + Dmat[1,S] + Dmat[2,S];

```

```

Common := objective_function(Alf,Sig);
Distance := Common - Unique;
end;

```

```

procedure cluster(var Smat:SimilarityMatrix; Mk,Mi:integer);
{determines common motor specs of clusters I and J, then
removes row J from the similarity matrix by marking element
0 with I of row J and all members of cluster J}

```

```

var
  I,J,II      : integer;
  Sij,Aij      : real;
  Dmat         : DisArray;

begin
  I := Mk; J := Mi;
  for II := 1 to NumAtts do
    begin
      Dmat[1,II] := Smat[I,II];
      Dmat[2,II] := Smat[J,II];
    end; Dmat[1,Q] := Smat[I,Q];
    Dmat[2,Q] := Smat[J,Q];
    Sij := sigma(Dmat); {assess excess functionality}
    Smat[I,S] := Smat[I,S]+Smat[J,S]+Sij; {sigma is cumulative}
    Aij := alpha(Dmat);
    Smat[I,f] := Smat[I,f]+Smat[J,f]+Aij; {alpha is cumulative}
    Smat[J,0] := I; {mark element 0 of row J, thus removing it }
    for II := 1 to NumItems do if Smat[II,0] = J then
      Smat[II,0] := I; { mark members of J, now I }
    end;
    Smat[I,Q] := Smat[I,Q] + Smat[J,Q];
    { total # of tanks in cluster I }
  end;
  for II := 1 to NumAtts do
    Smat[I,II] := max(Smat[I,II],Smat[J,II]);{common spec set}
  end;
end;

```

```

procedure display(Smat : SimilarityMatrix; Cost: real;
  var Flag : integer);

```

```

var
  I,J,C,II      : integer;
  Sfile          : text;
  Pfile          : file of Partition;
  Part           : Partition;

begin
  assign(Sfile,'comsol.dat'); rewrite(Sfile);
  assign(Pfile,'hiersol.dat'); rewrite(Pfile);
  Flag := 1;
  writeln(Sfile,'** HIERARCHICAL SOLUTION **');
  writeln(Sfile);
  C := 0;
  for I := 1 to NumItems do if
    Smat[I,0] = 0 then C := C + 1;
  end;
end;

```

```

writeln(Sfile,'Solution composed of ',C,' groups.',
        ' Net cost impact is ',Cost,'. ');
writeln(Sfile);
C := 1;
for I := 1 to NumItems do if Smat[I,0] = 0 then
begin
  for II := 1 to NumItems do Part.PartArray[II] := 0;
  II := 1;
  write(Sfile,'Group ',C,' members: ');
  write(Sfile,I,' ');
  Part.PartArray[II] := I;
  for J := 1 to NumItems do if Smat[J,0] = I then
begin
  write(Sfile,J,' ');
  II := II + 1;
  Part.PartArray[II] := J;
end;
  writeln(Sfile);
  write(Sfile,'Common specs < ');
  for J := 1 to NumAtts do write(Sfile,Smat[I,J]:8:2,' ');
  writeln(Sfile,'>');
  writeln(Sfile,Smat[I,Q]:8:2,' motors in cluster');
  C := C + 1;
  writeln(Sfile); writeln(Sfile);
  write(Pfile,Part);
end; close(Sfile); close(Pfile); end;

BEGIN
Flag := 0;
Cost := 4057.68; {Total "no commonality" cost}
Q := NumAtts + 1; {vector location of motor quantity}
S := NumAtts + 2; {vector location of excess functionality}
f := NumAtts + 3; {vector location of motor cost savings}
LRN := ln(LC)*1.4427; {learning curve exponent= ln(LC)/ln(2)}
assign(ComFile,'commtr.dat'); rewrite(ComFile);
getfile('motorcd.dat',MotorData);
{for I := 1 to NumItems do with MotorData[I] do
begin
  write(Mnum,' ',Mquan:2,' < ');
  for J := 1 to NumAtts do write(Mspecs[J]:8:3);
  write(' >');
  charv := readkey; writeln;
end;} for I := 1 to NumItems do
begin {load working data matrix}
  for J:=1 to NumAtts do
    SimMatrix[I,J]:=MotorData[I].Mspeccs[J];
  SimMatrix[I,Q] := MotorData[I].Mquan;
    {number of Motors per cluster}
  SimMatrix[I,0] := 0;
    {no motor is a member of another cluster}
  SimMatrix[I,S] := 0; {sigma is zero per cluster initially}
  SimMatrix[I,f] := 0; {alpha is zero per cluster initially}
end; NumClus := NumItems; {initially NumItems clusters}

```

```

While NumClus > 1 do
  begin
    MinDist := 1.0E30;
    for II := 1 to (NumItems - 1) do
      begin
        if SimMatrix[II,0] = 0 then
          begin
            (compute triangular similarity matrix)
            for J := 1 to (NumAtts+3) do
              DisMat[1,J] := SimMatrix[II,J];
            for I := (II+1) to NumItems do
              begin
                if SimMatrix[I,0] = 0 then
                  begin
                    for J := 1 to (NumAtts+3) do
                      DisMat[2,J] := SimMatrix[I,J];
                    Dist := Distance(DisMat);
                    if Dist < MinDist then
                      begin
                        MinDist := Dist;
                        Mink := II;
                        Mini := I;
                      end;
                    end;
                  end;
                end;
              end;
            end;
          end;
        Cost := Cost + MinDist;
        if (Flag=0) and (MinDist > 0) then
          Display(SimMatrix,(Cost-MinDist),Flag);
        cluster(SimMatrix,Mink,Mini);
        NumClus := NumClus - 1;
        writeln('Join cluster ',Mini,' to cluster ',Mink,
          ' at a distance of ',MinDist:8:2);
        write('Common specs < ');
        for J := 1 to NumAtts do
          write(SimMatrix[Mink,J]:8:3,' ');
        writeln('>');
        writeln(SimMatrix[Mink,Q]:3:0,' motors in cluster');
        writeln('Net cost impact (cumulative) = ',Cost:10:2);
        writeln(ComFile,'Join cluster ',Mini,' to cluster ',Mink,
          ' at a distance of ',MinDist:8:2);
        write(ComFile,'Common specs < ');
        for J := 1 to NumAtts do
          write(ComFile,SimMatrix[Mink,J]:8:3,' ');
        writeln(ComFile,'>');
        writeln(ComFile,SimMatrix[Mink,Q]:3:0,
          ' motors in cluster');
        writeln(Comfile,'Net cost impact (cumulative) = ',
          Cost:10:2);
        writeln(ComFile);
        ( write(' Press enter to continue =>');
          charv := readkey;) ( press key to continue)
        writeln;
        writeln;
      end;
    close(ComFile);
  END.

```

### C. Switching Program

program optimize;

uses printer, crt;

( This program attempts to improve on the hierarchcial clustering solution by evaluating possible switches between existing clusters. The partitioning data is input from the file HIERSOL.DAT. )

{ \$M 64000,0,655360 }

```
const                                (constants)
  K = 0.0;                            (K used for linear objective function)
  NumItems = 27;                      (number of different sizes of motors)
  NumAtts = 5;                        (number of attributes for each motor)
  LC = 0.85;                          (85% learning curve)

type
  Data = string;
  DataFile = text;
  Attarray = array[1..NumAtts] of real;
  ShString = string[35];
  DataRecord = record
    Mnum                : Data;
    Mquan               : integer;
    Mspecs              : Attarray;
  end;
  DataArray = array[1..NumItems] of DataRecord;
  SimilarityMatrix = array[1..NumItems,0..8] of real;
  DisArray = array[1..2,1..8] of real;      {8=NumAtts+3}
  Partition = record
    PartArray:array[1..NumItems] of integer;
  end;
  PartitionMatrix = array[1..NumItems,1..NumItems]
    of Integer;

var
  SimMatrix          : SimilarityMatrix;
  MotorNum           : ShString;
  MotorQuan          : integer;
  MotorSpecs         : Attarray;
  I,J,II,NumClus     : integer;
  NumMotor,C          : integer;
  Flag,P,L,M         : integer;
  Dist,Cost,NetCost   : real;
  Dfile,ComFile       : DataFile;
  DataLine           : Data;
  FileSpec            : string[80];
  MotorData           : DataArray;
```



```

MinCost                : real;
Mini,Mink               : integer;
charv                  : char;
Q,S,f                  : integer;
DisMat                 : DisArray;
Part                   : Partition;
Pfile,Ofile            : file of Partition;
Pmat,MinMat            : PartitionMatrix;
NumChanges             : integer;
LRN                    : real;

procedure decode(Line : Data; var Mnum : ShString;
                 Var Mquan:integer; var Mspecs : Attarray);
{unpacks the ASCII text lines and loads variables per line }

var
  Darray                : array[1..7] of ShString;(NumAtts+2)
  I,J,code,Marker       : integer;
  V                     : real;
  dummy                 : ShString;

begin
  J := 1;  Marker := 0;
  for I := 1 to length(Line) do
    {identify substrings by comma separators}
    if ((Line[I] = chr(44)) or (I = length(Line))) then
      begin { unpack substrings }
        if Line[I] = chr(44) then
          Darray[J] := copy(Line,Marker+1,I-Marker-1)
        else Darray[J] := copy(Line,Marker+1,I-Marker);
        Marker := I;
        J := J + 1;
      end;
  Mnum := Darray[1];                                {load variables}
  val(Darray[2],Mquan,code);
  for J := 3 to NumAtts+2 do
    begin
      dummy := Darray[J];
      if dummy[length(dummy)] = chr(32) then
        begin {write a '0' in Darray[J] if it has only blanks}
          dummy[length(dummy)] := chr(48);
          Darray[J] := dummy;
        end;
      val(Darray[J],V,code);
      if code <> 0 then
        begin
          writeln('Decoding error in string location ',code);
          writeln('of string ',J,' consisting of ',Darray[J]);
          halt;
        end
      end
    end
  end
end

```

```

    else Mspecs[J-2] := V;
    end; end;

procedure getfile(FileSpec : string;
                  var MotorData : DataArray);

var
    MotorNum          : ShString;
    MotorQuan         : integer;
    MotorSpecs        : Attarray;
    I,J,Recnum        : integer;
    Dfile              : text;
    Drecord            : string;

begin
    assign(Dfile, FileSpec); reset(Dfile);
    Recnum := 0;
    while not eof(Dfile) do
        begin
            Recnum := Recnum + 1;
            readln(Dfile,Drecord);
            decode(Drecord, MotorNum, MotorQuan, MotorSpecs);
            with MotorData[Recnum] do
                begin
                    Mnum := MotorNum;
                    Mquan := MotorQuan;
                    for J := 1 to NumAtts do Mspecs[J] := MotorSpecs[J];
                end;
            end;
        end;
    close(Dfile);
end;

function max(x,y:real):real;
begin
    if x < y then max := y else max := x;
end;

function min(x,y:real);real;
begin
    if x < y then min := x else min := y;
end;

function sigma(Dmat : DisArray) : real;
{ tally excess functionality to cluster I and J }

```

```

var
    QNi,QNj,sg                : real;
    I                          : integer;
    Maxi,Cj                   : array[1..NumAtts] of real;

begin
    sg := 0;
    QNi := Dmat[1,Q];
    QNj := Dmat[2,Q];
    Cj[1] := 0.5;      (500 dollars per watt)
    Cj[2] := 2.8;      (2800 dollars per pound)
    Cj[3] := 10.0;     (10,000 dollars per cubic foot)
    Cj[4] := 0.0;
    Cj[5] := 1.0;
    for I := 1 to NumAtts do
        Maxi[I] := max(Dmat[1,I],Dmat[2,I]);
        (define mean attribute vector above)
    for I:= 1 to NumAtts do
        sg := sg + QNi*Cj[I]*(abs(Dmat[1,I] - Maxi[I]))
            + QNj*Cj[I]*(abs(Dmat[2,I] - Maxi[I]));
        (sigma is increase in power requirement) sigma := sg;
    end;

function alpha(Dmat : DisArray):real;

var
    QNk,RD,Prod,LCS           : real;
    I,QN,QNii,QNji            : integer;
    QNi,QNj,LCSi,LCSj         : real;

begin
    QNi := Dmat[1,Q];  QNii := round(QNi);
    QNj := Dmat[2,Q];  QNji := round(QNj);
    QNk := QNi + QNj;
    QN := QNii + QNji;
    RD := min(Dmat[1,4],Dmat[2,4]);
    LCSi := 0.0;
    for I := 1 to QNii do LCSi := LCSi + exp(LRN*ln(I));
    LCSi := (QNi-LCSi)*Dmat[1,5];
    (learning curve savings in alpha i)

    LCSj := 0.0;
    for I := 1 to QNji do LCSj := LCSj + exp(LRN*ln(I));
    LCSj := (QNj-LCSj)*Dmat[2,5];
    (learning curve savings in alpha j)

    Prod := max(Dmat[1,5],Dmat[2,5]);
    LCS := 0.0;
    for I := 1 to QN do LCS := LCS + exp(LRN*ln(I));
    alpha := -(RD + Prod*(QNk - LCS) - LCSi - LCSj);
    (f1(N,S) is linear in N)
end;

```

```

function alphastar(Part1,Part2:Partition; Smat:SimMatrix;
                  N;integer):real;

var
  QNk, RD, Prod, LCS      : real;
  I, QN, QNii, QNji, J    : integer;
  QNi, QNj, LCSi, LCSj    : real;
  Max, Min                : real;

begin for I := 2 to NumItems do
  begin
    if Part1[1] > 0 then if Part1[I] > 0 then
      cluster(Smat,Part1[1],Part1[I]);
    if Part2[1] > 0 then if Part2[I] > 0 then
      cluster(Smat,Part2[1],Part2[I]);
    end; Max := 0.0; Min := 1.0E35; for I := 1 to NumItems
do if Smat[I,0] = Part1[1] then
  begin
    if Smat[I,4] > Max then Max := Smat[I,4];
    if Smat[I,4] < Min then Min := Smat[I,4];
    end; RD := Max - Min; QNk := Smat[Part1[1],Q] +
Smat[Part2[1],Q];
QN := round(QNk);
Max := 0.0;
for I := 1 to NumItems do if Smat[I,0] = Part1[1] then
  if Smat[I,5] > Max then Max := Smat[I,5];
LCSi := 0.0;
for I := 1 to QN do LCSi := LCSi + exp(LRN*ln(I));
LCSi := (QNk-LCSi)*Max;
      {max possible learning curve savings in switch}
Min := 1.0E35;
for I := 1 to NumItems do if Smat[I,0] = Part1[1] then
  begin
    {min possible learning curve savings left in cluster}
    QNi := Smat[I,Q]; QNii := round(QNi);
    LCSj := 0.0;
    for J := 1 to QNii do LCSj := LCSj + exp(LRN*ln(j));
    LCSj := (QNi-LCSj)*Smat[I,5];
    if LCSj < Min then Min := LCSj;
  end;
  Prod := Max - Min;
  QNi := Smat[Part1[1],0];
  QNii := round(QNi);
  LCSi := 0.0;
  for I := 1 to QNii do LCSi := LCSi + exp(LRN*ln(I));
  LCSi := (QNi-LCSi)*Smat[Part1[1],5];
      {learning curve savings in alpha i}
  QNj := Smat[Part2[1],0];
  QNji := round(QNj);
  LCSj := 0.0;
  for I := 1 to QNji do LCSj := LCSj + exp(LRN*ln(I));

```

```

LCSj := (QNj-LCSj)*Smat[Part2[1],5];
                                     {learning curve savings in alpha j}
Prod := Prod - LCSi - LCSj;
                                     {subtract original learning curve savings}
alphastar := RD + Prod;
end;

function objective_function(Alf,Sig:real):real;
begin
objective_function := Alf + Sig;
end;

function Distance(Dmat : DisArray) : real;
{assesses the inter-cluster distance between clusters I and J}

var
    Unique,Common,Alf,Sig      : real;

begin
Unique := objective_function(Dmat[1,f],Dmat[1,S])
          + objective_function(Dmat[2,f],Dmat[2,S]);
Alf := alpha(Dmat) + Dmat[1,f] + Dmat[2,f];
Sig := sigma(Dmat) + Dmat[1,S] + Dmat[2,S];
Common := objective_function(Alf,Sig);
Distance := Common - Unique;
end;

procedure cluster(var Smat:SimilarityMatrix; Mk,Mi:integer);
{determines common motor specs of clusters I and J, then
removes row J from the similarity matrix by marking element
0 with I of row J and all members of cluster J}

var
    I,J,II      : integer;
    Sij,Aij      : real;
    Dmat         : DisArray;

begin
I := Mk; J := Mi;
for II := 1 to NumAtts+3 do
begin
Dmat[1,II] := Smat[I,II];
Dmat[2,II] := Smat[J,II];
end; Sij := sigma(Dmat); {assess excess functionality}
Smat[I,S] := Smat[I,S] + Smat[J,S] + Sij;
                                     {sigma is cumulative}
Aij := alpha(Dmat);

```

```

Smat[I,f] := Smat[I,f] + Smat[J,f] + Aij;
                                     {alpha is cumulative}
Smat[J,0] := I;
      { mark element 0 of row J, thus removing it }
for II := 1 to NumItems do if Smat[II,0]=J then
  Smat[II,0] := I;      { mark members of J, now I }
Smat[I,Q] := Smat[I,Q] + Smat[J,Q];
      { total # of motors in cluster I }
for II := 1 to NumAtts do
  Smat[I,II] := max(Smat[I,II],Smat[J,II]){common spec set}
end;

```

```

function Scost(Pmat : PartitionMatrix;
               Smat : SimilarityMatrix;
               Mdat : DataArray):real;

```

```

var
  I,J,N,C      : integer;
  Ccost,Netcost : real;

begin
  NetCost := 0.0;
  C := 0; for
  J := 1 to NumItems do if Pmat[J,1] > 0 then
    begin
      C := C + 1;
      N := Pmat[J,1];
      for I := 2 to NumItems do if Pmat[J,I] > 0 then
        cluster(Smat,N,Pmat[J,I]);
        Ccost := Smat[N,f] + Smat[N,S];
        {alpha and sigma calculated in cluster}
        NetCost := NetCost + Ccost;
      end;
    end;
  Scost := NetCost + 4057.68;
end;

```

```

function Scost_w_output(Pmat : PartitionMatrix;
                        Smat : SimilarityMatrix;
                        Mdat : DataArray):real;

```

```

var
  I,J,N,C      : integer;
  Ccost,Ntcost : real;

begin
  NtCost := 0.0;
  C := 0;
  for J := 1 to NumItems do if Pmat[J,1] > 0 then

```

```

begin
  C := C + 1;
  N := Pmat[J,1];
  for I := 2 to NumItems do if Pmat[J,I] > 0 then
    cluster(Smat,N,Pmat[J,I]);
    Ccost := Smat[N,f] + Smat[N,S];
    {alpha and sigma calculated in cluster}
    writeln('Cluster ',C:2,' cost = ',Ccost:1:2);
    writeln;
    NtCost := NtCost + Ccost;
  end;
  Scost_w_output := NtCost + 4057.68;
end;

procedure find(Pmat : PartitionMatrix; P,R : integer;
               var I,J : integer);

begin
  J := 0; I := R;
  if R = 0 then { search for P over entire matrix }
    begin
      I := 1;
      repeat
        J := J + 1;
        if Pmat[I,J] = 0 then
          begin
            I := I + 1;
            J := 1;
          end;
        if Pmat[I,1] = 0 then
          begin
            writeln('No match in partition array.'
                  , ' Pmat[' , I , ':' , J , ']' );
            halt;
          end;
        until Pmat[I,J] = P;
      end else { search for first 0 in row R }
        repeat
          J := J + 1;
          if J > NumItems then
            begin
              writeln('No match in partition array 2');
              halt;
            end;
          until Pmat[I,J] = P;
        end;
    end;
end;

```

```

procedure display(Pmat : PartitionMatrix;
                  Smat : SimilarityMatrix;
                  Cost : real);

var
  I,J,C,II      : integer;
  Sfile         : text;
  Ofile         : file of Partition;
  Part          : Partition;

begin
  assign(Ofile,'swsol.dat'); rewrite(Ofile);
  for I := 1 to NumItems do
    begin
      for J := 1 to NumItems do Part.PartArray[J] := Pmat[I,J];
      write(Ofile,Part);  (save optimum partition matrix)
    end;
  close(Ofile);
  assign(Sfile,'optsol.dat'); rewrite(Sfile);
  for J := 1 to NumItems do if Pmat[J,1] > 0 then
    begin
      II := Pmat[J,1];
      for I := 2 to NumItems do if Pmat[J,I] > 0 then
        cluster(Smat,II,Pmat[J,I]);
      end;
    writeln(Sfile,'** FINAL SOLUTION **');
    writeln(Sfile);
    C := 0;
    for I := 1 to NumItems do if Smat[I,0] = 0 then C := C + 1;
    writeln(Sfile,'Solution composed of ',C,' groups.',
              ' Net cost impact is ',Cost:10:2,'. ');
    writeln(Sfile);
    C := 1;
    for I := 1 to NumItems do if Smat[I,0] = 0 then
      begin
        write(Sfile,'Group ',C,' members: ');
        write(Sfile,I,' ');
        for J := 1 to NumItems do if Smat[J,0] = I then
          write(Sfile,J,' ');
        writeln(Sfile);
        write(Sfile,'Common specs < ');
        for J := 1 to NumAtts do write(Sfile,Smat[I,J]:8:2,' ');
        writeln(Sfile,'>');
        writeln(Sfile,Smat[I,Q]:8:2,' motors in cluster');
        C := C + 1;
        writeln(Sfile); writeln(Sfile);
      end;
    close(Sfile);
  end;
end;

```



```

BEGIN
Q := NumAtts + 1;
S := NumAtts + 2;
f := NumAtts + 3;
LRN := ln(LC)*1.4427; {learning curve exponent=ln(LC)/ln(2)}
getfile('motorcd.dat',MotorData);
assign(Pfile,'hiersol.dat'); reset(PFile);
{for I := 1 to NumItems do with MotorData[I] do
begin
write(Mnum,' ',Mquan,' < ');
for J := 1 to NumAtts do write(Mspecs[J]:8:3);
write(' >');
charv := readkey; writeln;
end;} for I := 1 to NumItems do
begin
for J := 1 to NumAtts do
SimMatrix[I,J] := MotorData[I].Mspects[J];
SimMatrix[I,Q] := MotorData[I].Mquan;
{number of Motors per cluster}
SimMatrix[I,0] := 0.0;
{no motor is a member of another cluster}
SimMatrix[I,S] := 0.0;
{sigma is zero per cluster initially}
SimMatrix[I,f] := 0.0;
{alpha is zero per cluster initially}
end; NetCost := 0.0; Cost := 0.0; writeln; for I := 1 to
NumItems do for J := 1 to NumItems do
Pmat[I,J] := 0;
for J := 1 to NumItems do if not eof(Pfile) then
begin
for I := 1 to NumItems do Part.PartArray[I] := 0;
read(Pfile,Part);
for I := 1 to NumItems do Pmat[J,I] := Part.PartArray[I];
end; NetCost := Scost_w_output(Pmat,SimMatrix,MotorData);
writeln;
writeln('Total cost is ',NetCost:1:2);
write('any key to continue =>'); charv := readkey; writeln;
close(Pfile);
repeat
NumChanges := 0;
for I := 1 to NumItems do for J := 1 to NumItems do
MinMat[I,J] := Pmat[I,J];
for P := 1 to NumItems do
begin
find(MinMat,P,0,NumClus,NumMotor);
find(MinMat,0,NumClus,L,M);
MinMat[NumClus,NumMotor] := MinMat[NumClus,M-1];
MinMat[NumClus,M-1] := 0;
for II := 1 to NumItems do {cluster number = II}
if MinMat[II,1] > 0 then

```

```

begin
  find(MinMat,0,II,I,J); {first zero in row II}
  MinMat[I,J] := P; make that element motor P
  MinCost := Scost(MinMat,SimMatrix,MotorData);
  if MinCost < NetCost then
    begin
      (save new lower cost partition)
      for L := 1 to NumItems do
        for M:=1 to NumItems do
          Pmat[L,M]:=MinMat[L,M];
        NetCost := MinCost;
        NumChanges := NumChanges + 1;
      end;
      MinMat[I,J] := 0;
    end;
    find(MinMat,0,NumClus,L,M); {reset matrix}
    MinMat[NumClus,M] := MinMat[NumClus,NumMotor];
    MinMat[NumClus,NumMotor] := P;
  end;
writeln('Pass complete. ',NumChanges,' change(s) made.');
```

writeln;

until NumChanges = 0;

Display(Pmat,SimMatrix,NetCost);

END.

#### D. Hierarchical Clustering Output

Join cluster 17 to cluster 16 at a distance of -138.87  
Common specs < 510.000 5.865 0.068 118.290 56.210 >  
6 motors in cluster  
Net cost impact (cumulative) = 3918.81

Join cluster 14 to cluster 13 at a distance of -85.62  
Common specs < 300.000 4.200 0.049 94.840 45.070 >  
6 motors in cluster  
Net cost impact (cumulative) = 3833.18

Join cluster 9 to cluster 8 at a distance of -58.55  
Common specs < 32.000 1.408 0.016 46.020 21.870 >  
12 motors in cluster  
Net cost impact (cumulative) = 3774.63

Join cluster 23 to cluster 22 at a distance of -50.58  
Common specs < 67.000 0.858 0.010 33.160 15.760 >  
30 motors in cluster  
Net cost impact (cumulative) = 3724.05

Join cluster 15 to cluster 11 at a distance of -49.27  
Common specs < 350.000 5.880 0.068 118.490 56.310 >  
6 motors in cluster  
Net cost impact (cumulative) = 3674.78

Join cluster 27 to cluster 18 at a distance of -47.41  
Common specs < 200.000 1.800 0.021 54.140 25.730 >  
8 motors in cluster  
Net cost impact (cumulative) = 3627.37

Join cluster 21 to cluster 20 at a distance of -39.52  
Common specs < 80.000 0.869 0.010 33.440 15.890 >  
12 motors in cluster  
Net cost impact (cumulative) = 3587.85

Join cluster 8 to cluster 7 at a distance of -36.90  
Common specs < 32.000 1.408 0.016 46.020 21.870 >  
14 motors in cluster  
Net cost impact (cumulative) = 3550.94

Join cluster 4 to cluster 2 at a distance of -30.04  
Common specs < 6.800 0.554 0.006 24.830 11.800 >  
8 motors in cluster  
Net cost impact (cumulative) = 3520.90

Join cluster 26 to cluster 24 at a distance of -28.60  
Common specs < 13.000 0.425 0.005 20.840 9.900 >  
44 motors in cluster  
Net cost impact (cumulative) = 3492.30

Join cluster 25 to cluster 24 at a distance of -28.88  
 Common specs < 13.000 0.425 0.005 20.840 9.900 >  
 46 motors in cluster  
 Net cost impact (cumulative) = 3463.42

Join cluster 6 to cluster 2 at a distance of -25.64  
 Common specs < 10.000 0.600 0.007 26.170 12.440 >  
 16 motors in cluster  
 Net cost impact (cumulative) = 3437.78

Join cluster 5 to cluster 2 at a distance of -20.44  
 Common specs < 10.000 0.679 0.008 28.410 13.500 >  
 18 motors in cluster  
 Net cost impact (cumulative) = 3417.34

Join cluster 24 to cluster 1 at a distance of 16.48  
 Common specs < 13.000 0.425 0.005 20.840 9.900 >  
 58 motors in cluster  
 Net cost impact (cumulative) = 3433.82

Join cluster 3 to cluster 2 at a distance of 19.31  
 Common specs < 10.000 0.679 0.008 28.410 13.500 >  
 40 motors in cluster  
 Net cost impact (cumulative) = 3453.13

Join cluster 20 to cluster 19 at a distance of 66.23  
 Common specs < 98.000 0.960 0.011 35.720 16.980 >  
 16 motors in cluster  
 Net cost impact (cumulative) = 3519.36

Join cluster 13 to cluster 11 at a distance of 73.49  
 Common specs < 350.000 5.880 0.068 118.490 56.310 >  
 12 motors in cluster  
 Net cost impact (cumulative) = 3592.85

Join cluster 2 to cluster 1 at a distance of 90.80  
 Common specs < 13.000 0.679 0.008 28.410 13.500 >  
 98 motors in cluster  
 Net cost impact (cumulative) = 3683.65

Join cluster 19 to cluster 10 at a distance of 202.92  
 Common specs < 98.000 2.448 0.028 66.360 31.530 >  
 20 motors in cluster  
 Net cost impact (cumulative) = 3886.57

Join cluster 18 to cluster 12 at a distance of 214.16  
 Common specs < 250.000 3.500 0.041 84.060 39.950 >  
 15 motors in cluster  
 Net cost impact (cumulative) = 4100.73

Join cluster 22 to cluster 7 at a distance of 291.31  
 Common specs < 67.000 1.408 0.016 46.020 21.870 >  
 44 motors in cluster  
 Net cost impact (cumulative) = 4392.04

Join cluster 16 to cluster 11 at a distance of 756.44  
 Common specs < 510.000 5.880 0.068 118.490 56.310 >  
 18 motors in cluster  
 Net cost impact (cumulative) = 5148.48

Join cluster 10 to cluster 7 at a distance of 848.72  
 Common specs < 98.000 2.448 0.028 66.360 31.530 >  
 64 motors in cluster  
 Net cost impact (cumulative) = 5997.20

Join cluster 12 to cluster 11 at a distance of 1971.09  
 Common specs < 510.000 5.880 0.068 118.490 56.310 >  
 33 motors in cluster  
 Net cost impact (cumulative) = 7968.30

Join cluster 7 to cluster 1 at a distance of 5088.84  
 Common specs < 98.000 2.448 0.028 66.360 31.530 >  
 162 motors in cluster  
 Net cost impact (cumulative) = 13057.13

Join cluster 11 to cluster 1 at a distance of 36015.72  
 Common specs < 510.000 5.880 0.068 118.490 56.310 >  
 195 motors in cluster  
 Net cost impact (cumulative) = 49072.85

**\*\* HIERARCHICAL SOLUTION \*\***

Solution composed of 14 groups. Net cost impact is 3417.34

Group 1 members: 1

Common specs <	3.50	0.35	0.00	18.32	8.71 >
12.00 motors in cluster					

Group 2 members: 2 4 5 6

Common specs <	10.00	0.68	0.01	28.41	13.50 >
18.00 motors in cluster					

Group 3 members: 3

Common specs <	6.50	0.01	0.00	26.80	12.73 >
22.00 motors in cluster					

Group 4 members: 7 8 9

Common specs <	32.00	1.41	0.02	46.02	21.87 >
14.00 motors in cluster					

Group 5 members: 10

Common specs <	68.00	2.45	0.03	66.36	31.53 >
4.00 motors in cluster					

Group 6 members: 11 15

Common specs <	350.00	5.88	0.07	118.49	56.31 >
6.00 motors in cluster					

Group 7 members: 12

Common specs <	250.00	3.50	0.04	84.06	39.95 >
7.00 motors in cluster					

Group 8 members: 13 14

Common specs <	300.00	4.20	0.05	94.84	45.07 >
6.00 motors in cluster					

Group 9 members: 16 17

Common specs <	510.00	5.86	0.07	118.29	56.21 >
6.00 motors in cluster					

Group 10 members: 18 27				
Common specs < 200.00	1.80	0.02	54.14	25.73 >
8.00 motors in cluster				
Group 11 members: 19				
Common specs < 98.00	0.96	0.01	35.72	16.98 >
4.00 motors in cluster				
Group 12 members: 20 21				
Common specs < 80.00	0.87	0.01	33.44	15.89 >
12.00 motors in cluster				
Group 13 members: 22 23				
Common specs < 67.00	0.86	0.01	33.16	15.76 >
30.00 motors in cluster				
Group 14 members: 24 25 26				
Common specs < 13.00	0.42	0.01	20.84	9.90 >
46.00 motors in cluster				

## E. Switching Program Output

### \*\* FINAL SOLUTION \*\*

Solution composed of 14 groups. Net cost impact is 3417.34.

Group 1 members: 1

Common specs <	3.50	0.35	0.00	18.32	8.71 >
12.00 motors in cluster					

Group 2 members: 2 4 5 6

Common specs <	10.00	0.68	0.01	28.41	13.50 >
18.00 motors in cluster					

Group 3 members: 3

Common specs <	6.50	0.01	0.00	26.80	12.73 >
22.00 motors in cluster					

Group 4 members: 7 8 9

Common specs <	32.00	1.41	0.02	46.02	21.87 >
14.00 motors in cluster					

Group 5 members: 10

Common specs <	68.00	2.45	0.03	66.36	31.53 >
4.00 motors in cluster					

Group 6 members: 11 15

Common specs <	350.00	5.88	0.07	118.49	56.31 >
6.00 motors in cluster					

Group 7 members: 12

Common specs <	250.00	3.50	0.04	84.06	39.95 >
7.00 motors in cluster					

Group 8 members: 13 14

Common specs <	300.00	4.20	0.05	94.84	45.07 >
6.00 motors in cluster					

Group 9 members: 16 17

Common specs <	510.00	5.86	0.07	118.29	56.21 >
6.00 motors in cluster					



Group 10 members: 18 27				
Common specs <	200.00	1.80	0.02	54.14
8.00 motors in cluster				25.73 >
Group 11 members: 19				
Common specs <	98.00	0.96	0.01	35.72
4.00 motors in cluster				16.98 >
Group 12 members: 20 21				
Common specs <	80.00	0.87	0.01	33.44
12.00 motors in cluster				15.89 >
Group 13 members: 22 23				
Common specs <	67.00	0.86	0.01	33.16
30.00 motors in cluster				15.76 >
Group 14 members: 24 25 26				
Common specs <	13.00	0.42	0.00	20.84
46.00 motors in cluster				9.90 >



APPENDIX C. APPLICATION OF SUFFICIENCY CONDITIONS TO PHASE  
C/D COMMONALITY ANALYSIS CLUSTERING SOLUTION

Employing the sufficiency condition of equation (13) for Z-minimal partitions, a total of 91 inequalities may be defined for the 14 clusters given in Table 18 as follows:

1.  $\sigma_1 + \sigma_2 + \text{Maximum}\{\alpha^*(1), \alpha^*(2)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=1, v=2,4,5,6$
2.  $\sigma_1 + \sigma_3 + \text{Maximum}\{\alpha^*(1), \alpha^*(3)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=1, v=3$
3.  $\sigma_1 + \sigma_7 + \text{Maximum}\{\alpha^*(1), \alpha^*(7)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=1, v=7,8,9$
- .
- .
- .
14.  $\sigma_2 + \sigma_3 + \text{Maximum}\{\alpha^*(2), \alpha^*(3)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=2,4,5,6, v=3$
15.  $\sigma_2 + \sigma_7 + \text{Maximum}\{\alpha^*(2), \alpha^*(7)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=2,4,5,6, v=7,8,9$
16.  $\sigma_2 + \sigma_{10} + \text{Maximum}\{\alpha^*(2), \alpha^*(10)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=2,4,5,6, v=10$
- .
- .
- .
91.  $\sigma_{22} + \sigma_{24} + \text{Maximum}\{\alpha^*(22), \alpha^*(24)\} \leq \text{Minimum}\{\delta(u,v)\},$   
 $u=22,23, v=24,25,26$

Consider inequality number 1. With  $\sigma_1 = 0$ ,  $\sigma_2 = 44.8$ ,  $\alpha^*(1) = 0$ , and  $\alpha^*(2) = 55.2$ , the minimum  $\delta(u,v)$  is  $\delta(1,4) = 56.4$ . Since  $44.8 + 55.2 > 56.4$ , this inequality does not hold, and the minimum partition of  $\{1,2,4,5,6\}$  into two groups must be determined by complete enumeration if all other inequalities hold. However, now consider that inequalities number 13 and 25 also do not hold; inequality number 13 concerns clusters 1 and 24 while inequality number 25 concerns clusters 2 and 24. Thus, the minimum partition of  $\{1,2,4,5,6,24,25,26\}$  into three groups must be determined by complete enumeration if all other inequalities hold.

Of the first 25 inequalities, inequalities number 1, 13, 14, 15, 17, 19, 23, 24, and 25 do not hold. Thus, the optimal partition of  $\{1,2,3,4,5,6,7,8,9,11,13,14,15,20,21,22,23,24,25\}$  into 9 groups must be determined by enumeration. The number of partitions of 19 objects into 9 groups is greater than  $1.14 \times 10^{12}$ . Thus, although

the validity of inequality number 2 serves to reduce the solution space by almost  $3.0 \times 10^{10}$  partitions, the impact is negligible.

While the remainder of inequalities numbered 1 through 25 hold to reduce the number of partitions which must be examined in some measure, inequalities numbered 26 through 91 have yet to be examined. Thus, at this point it is intuitive that the partition of Table 18 cannot be verified as Z-minimal employing the sufficiency conditions in a manual branch-and-bound manner. A computer program would be required to determine the number of partitions remaining after all inequalities were examined, thus establishing whether enumeration of partitions is feasible.

## REFERENCES

1. Blanchard, B. S. and Fabrycky, W. J.: Systems Engineering and Analysis. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.
2. Thomas, L. D.: SCAT: A Commonality Analysis Model. Proceedings of the Workshop on Commonality in Space Systems, Johnson Space Center, October 1986, pp. 264-275.
3. Waiss, R. D.: Cost Reduction on Large Space Systems Through Commonality. Paper presented at the AIAA 25th Ann. Aerospace Sci. Meeting, Reno, Nevada, January 1987.
4. Powell, L. E. and Beam, E. E.: Commonality Analysis for the Space Station Common Module. Proc. 36th International Aerospace Federation (IAF) Congress, Stockholm, October 1985.
5. "System Commonality Analysis Tool (SCAT) User's and Programmer's Manuals." Prepared for the National Aeronautics and Space Administration by Boeing Aerospace Company, Seattle, Washington, Contract NAS8-36413, Document Number D483-10064, May 1987.
6. Duran, B. S. and Odell, P.: Cluster Analysis - A Survey. Springer-Verlag, New York, 1974.
7. Everitt, B. S.: Unresolved Problems in Cluster Analysis. Biometrics, Vol. 35, 1979, pp. 169-181.
8. Burbidge, J. L.: Production Flow Analysis. Production Engineer, Vol. 42, No. 12, 1963, pp. 742-752.
9. McAuley, J.: Machine Grouping for Efficient Production. Production Engineer, Vol. 51, No. 2, 1972, pp. 53-57.
10. King, J. R.: Machine-Component Group Formation in Group Technology. Omega, Vol. 8, No. 2, 1980, pp. 193-199.
11. Kusiak, A. and Chow, W. S.: Efficient Solving of the Group Technology Problem. Journal of Manufacturing Systems, Vol. 6, No. 2, 1987, pp. 117-124.
12. Seifoddini, H. and Wolfe, P. M.: Application of the Similarity Coefficient Method in Group Technology. IIE Transactions, Vol. 18, No. 3, 1986, pp. 271-277.
13. McCormick, W. T., Schweitzer, P. J., and White, T. W.: Problem Decomposition and Data Reorganization by a Clustering Technique. Operations Research, Vol. 20, 1972, pp. 993-1009.
14. Perrin, T. M.: A Classification System for Tactical Weapons. Unpublished Master's Thesis, The University of Alabama in Huntsville, 1987.
15. Yeager, D. P.: A Formalization of the Commonality Analysis Problem and Some Partial Solutions. Paper submitted to Mathematics of Operations Research.

16. Mojena, R.: Hierarchical Grouping Methods and Stopping Rules: An Evaluation. *The Computer Journal*, Vol. 20, No. 4, 1977, pp. 359-363.
17. Valev, V.: Set Partition Principles. *Transactions of the Ninth Prague Conf. on Information Theory, Statistical Decision Functions, and Random Processes*, Prague, Czechoslovakia, June 1982, pp. 251-256.
18. Blashfield, R. K.: Mixture Model Tests of Cluster Analysis: Accuracy of Four Agglomerative Hierarchical Methods. *Psychological Bulletin*, Vol. 83, 1976, pp. 377-388.
19. Milligan, G. W., Soon, S. C., and Sokol, L. M.: The Effect of Cluster Size, Dimensionality, and Number of Clusters on Recovery of True Cluster Structure. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-5(1), 1983, pp. 40-47.
20. Ward, J. H.: Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, Vol. 58, 1963, pp. 236-244.
21. Zupan, J.: *Clustering of Large Data Sets*. Research Studies Press of John Wiley and Sons, New York, 1982.
22. Romesburg, H. C.: *Cluster Analysis for Researchers*. Lifetime Learning Publications, Belmont, California, 1984.
23. Johnson, S. C.: Hierarchical Clustering Schemes. *Psychometrika*, Vol. 32, 1967, pp. 241-254.
24. Jensen, R. E.: A Dynamic Programming Algorithm for Cluster Analysis. *Operations Research*, Vol. 17, 1969, pp. 1034-1057.
25. Anderberg, M. R.: *Cluster Analysis for Applications*. Academic Press, New York, 1973.
26. Thomas, L. D.: Commonality Analysis Using Clustering Methods. A Paper submitted to *IEEE Trans. on Systems, Man, and Cybernetics*.
27. "Project Management Handbook," National Aeronautics and Space Administration, Document Number MMI-7120.2, August 1985.
28. Ostrofsky, B.: *Design, Planning, and Development Methodology*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.
29. Personal communication with C. R. Darwin, Director, Program Development Directorate, NASA-Marshall Space Flight Center, Huntsville, Alabama, September 1988.
30. "Space Station Interface Commonality Study: Final Report." Prepared for the National Aeronautics and Space Administration by Spectra Research Systems, Huntsville, Alabama, Contract Number H-62325B), Document Number SRS/SE-TR83-113, April 1983.
31. Personal communication with J. M. McMillion, Director, Systems Analysis and Integration Laboratory, NASA-Marshall Space Flight Center, Huntsville, Alabama, September 1988.

32. Personal communication with E. R. Tanner, Manager, Space Station Projects Office, NASA-Marshall Space Flight Center, Huntsville, Alabama, September 1988.
33. Engleman, L. and Hartigan, J. A.: Percentage Points of a Test for Clusters. American Statistical Association Journal, Vol. 64, 1969, pp. 1647-1648.
34. Jardine, N. and Sibson, R.: The Construction of Hierarchic and Non-Hierarchic Classifications. The Computer Journal, Vol. 11, 1968, pp. 177-184.

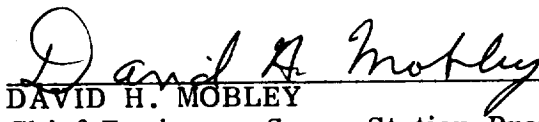
C-2

## APPROVAL

### A METHODOLOGY FOR COMMONALITY ANALYSIS, WITH APPLICATIONS TO SELECTED SPACE STATION SYSTEMS

By L. Dale Thomas

The information in this report has been reviewed for technical content. Review of any information concerning Department of Defense or nuclear energy activities or programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

  
\_\_\_\_\_  
DAVID H. MOBLEY  
Chief Engineer, Space Station Projects Office